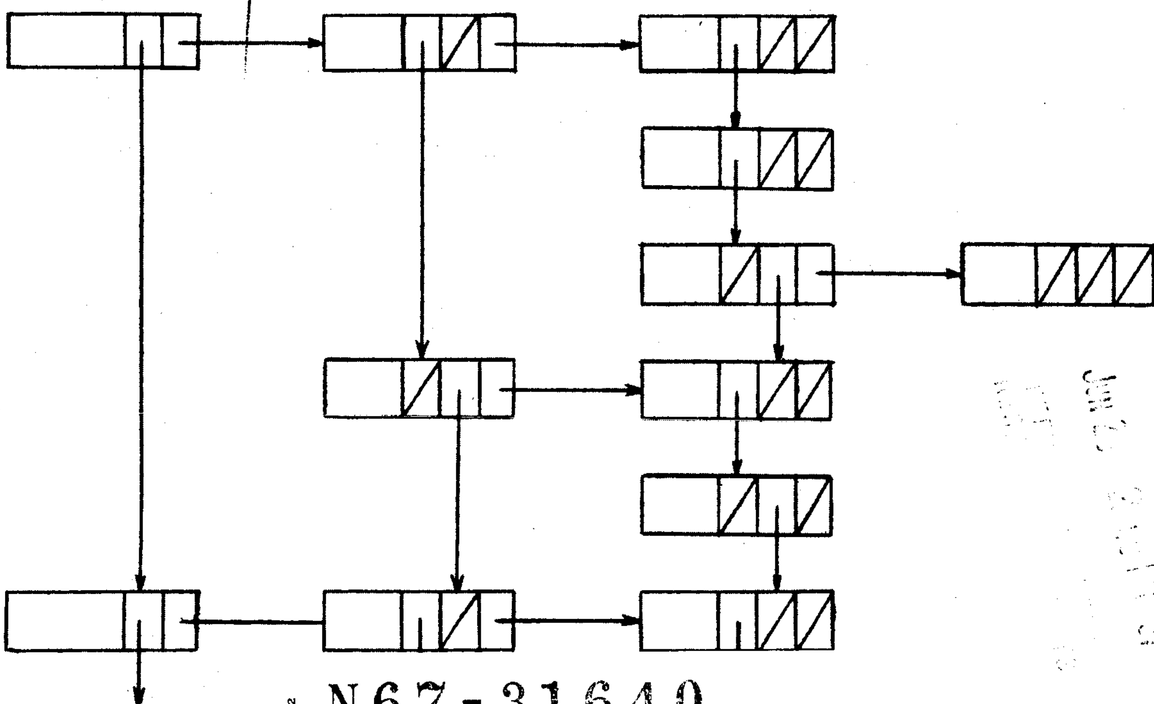
SPATIAL DATA SYSTEMS:

SYSTEMS CONSIDERATIONS

Kenneth J. Dueker



Geography Branch
Office of Naval Research
Task No. 389-143
Contract Nonr 1228(37)

Technical Report No. 5
Urban and Transportation Information Systems

Department of Geography
Northwestern University

SPATIAL DATA SYSTEMS: SYSTEMS CONSIDERATIONS

by

Kenneth J. Dueker

TECHNICAL REPORT NO. 5

of

ONR Task No. 389-143
Contract Nonr 1228(37)

OFFICE OF NAVAL RESEARCH

GEOGRAPHY BRANCH

Department of Geography
Northwestern University
Evanston, Illinois

December 1966

Funding Provided by
Natural Resources Program
Office of Space Sciences and Applications
National Aeronautics and Space Administration

PREFACE

This report provides a limited explication of current needs for classifying and organizing spatial data for use in urban and transportation planning. In addition, requirements and methods for handling spatial data are explored. Mr. Dueker emphasizes the dual need for data organization methods and data handling capabilities, as a requisite for utilization of data acquired from remote sensors mounted on earth orbital platforms. This work provides a basis for examination of some problems of integrating remote sensors into a viable geographic information system.

SPATIAL DATA SYSTEMS

by

Kenneth J. Dueker

ABSTRACT

Spatial data systems are concerned with the organization, handling, and retrieval of data whose spatial position is of concern. Spatial data are of particular concern in urban and transportation planning. In these fields considerable attention is given to spatial and temporal variations of data.

The three volume report presents a discussion of concepts and techniques that are essential in moving towards flexible and responsive urban information systems. The following areas are emphasized:

1. Explication of terms associated with spatial data.
2. Discussion of means of organizing spatial data for flexible and efficient retrieval.
3. Investigation of data handling capabilities for organizing and manipulating spatial data.
4. Presentation of these topics in a tutorial form, conceivably to serve as a text where none presently exist.

The greater speeds and storage capacities of newer computers requires new concepts of data organization and new means to create and access these more complex data structures. Of particular concern in urban and transportation planning are needs to link separately collected data that relate to the same phenomena or spatial locations, and a need for user-oriented data

handling capabilities. These needs are explored and recommendations are made.

Volume 1 contains the Summary and Conclusions, Introduction, Data Handling Capabilities for Supplying the Demands for Urban Data, Nature of Spatial-Temporal Data, and Questions and Queries.

Volume 2 deals with a Language to Query Spatial-Temporal Data, a Directory of Spatial-Temporal Data, and Hardware-Software Configurations.

Volume 3 deals with Entity Linkage, Organization of Data for Retrieval, Search Efficiencies, Point in Polygon Procedures and Retrieval from Digitized Imagery.

Part III

SYSTEMS CONSIDERATIONS

Chapter 5. LANGUAGE TO QUERY SPATIAL-TEMPORAL DATA

Computer programs are detailed instructions written to adapt a completely general purpose machine to a specific problem or application. To solve a problem using a computer, one must write a sequence of instructions, called a program, to control the arithmetic and logical circuitry of the machine. Another type of computer program, called "software", is designed and written by computer specialists to provide users with more than just an empty machine. Software may be used for building, controlling and modifying the complex sequences of problem solving procedures required in his computer applications. Essentially, the motivations for the development of software have been to improve the ease with which applications programs can be written and problems solved, and to reduce the cost of constructing, modifying, and executing applications programs and solving problems.

Handling Spatial Data

Urban and transportation planning analyses require software to ease solving problems and the manipulation of spatial data. Handling spatial data has two sets of facets. One set relates to characteristics of the handlers or users, and the other relates to characteristics of spatial data.

User requirements are of five types. These are:

1. To simplify access to the computer by the user. User-orientation permits control of a problem-solver or data handling capability by persons other than those with a specific programming background.

2. To provide a flexible and powerful language in which to define problems. A high degree of flexibility is necessary to handle diverse problems relating to urban and transportation planning. A powerful system is also necessary to permit a maximum processing capability through a minimum of programming or formal specification of commands.

3. To bring about automation in the performance of tasks. The user should not be burdened with the task of programming the details of often-used procedures, or documentation of inputs and outputs.

4. To link relatively independent program functions. Different procedures should operate in concert. The outputs from one program function should be acceptable by another. Natural and reasonably automatic communication of data across programs or operations is essential in urban and transportation planning applications.

5. To design a system that can be readily expanded to handle additional problems in urban and transportation planning. Any software system should be designed for growth and improvements.

Requirements that relate to using spatial data for urban and transportation planning are of three types. These are:

1. To simplify access to data by locational criteria. Urban and transportation planning analyses often require selection of data by spatial location. This problem is twofold and relate, a) to the programming system and b) to the organization of data into separate files or lists for different spatial areas. With respect to the programming system, the selection process should be easily defined and efficient.

2. To simplify access to data by temporal criteria. In a way similar to spatial location criteria, the programming system should permit an easily defined and efficient selection of data items by temporal characteristics.

3. To retrieve or select related data from more than one file. To select different data related to a person or several items of data relating to a location is a desirable capability. Depending upon how data are organized, a programming system must be capable of matching entities from different files or searching data stored on chain structures.

These requirements for handling spatial data can only be partially met by existing data handling capabilities, such as those described in Part II, Chapter 2. The only system, of those reviewed in Chapter 2* that meets the user requirements is GIS (Generalized Information System). However, at the time of this writing it is not fully operational. In addition, it will only be available to IBM System/360 users. Although GIS does not fully meet the requirements relating to spatial-temporal data, the system apparently could be augmented to meet these needs of selecting data by spatial location. Even with systems such as GIS, there is a need for experimentation with specially designed systems for handling spatial data. Except for Span, and to some extent Mark III, the systems are not oriented to urban and transportation planning. However, these two are very much hardware dependent and confined to data stored on serial files.

What is needed is a user-oriented programming language for expressing problems relating to manipulation of spatial data. The objective is to give the planner a tool for communicating, in something near his own language, with the computer. Such a system should be designed independent of hardware to enable more widespread implementation.

The emphasis upon user-orientation excludes the general purpose programming languages. General or procedure-oriented programming systems such as Fortran, Algol and Cobol are not considered here because of their orientation to more universal applications. Greater skill and experience is required to use these systems than can be demanded for users in urban and transportation planning.

Similarly, list processors, such as Lisp, IPL V, and Slip are not suitable because they require specialized programming skills and are not oriented to handling large files of spatial data. Systems such as Simscript

*Span, Mark III, Colingo, IDS, GIS and On-line Data Management.

that are designed for other problems are also not suitable. Even though Simscript has general programming and list processing capabilities, its documentation and orientation is foreign to urban and transportation planners. Urban and transportation planners should not have to rely upon the tools of others, as their needs are sufficient to demand their own tools.

Introduction to Quest

The Quest (QUEry of Spatial-Temporal Data) language is proposed to facilitate the manipulation and retrieval of spatially distributed data. Quest would permit the flexible expression of instructions in user-oriented terms. The user-oriented terms used by Quest relate to the definitions concerning data and the principal dimensions, which were introduced in Chapter 3.

Although the Quest language is relevant to the user, it is not a highly sophisticated language.* Quest is a simplified and highly stylized language for communicating with the computer. Therefore, Quest is at a lower level of syntactical complexity and is more highly formated and stylized than natural English language.

Quest is designed to enable accessing data in unanticipated ways; in ways that cut across any preconceived organization of these data. The purpose of Quest is to enable a user to approach data in novel and unanticipated ways and to elicit data relevant to his needs.

Quest does not include a complex processing capability. Rather, Quest output provides input data for powerful algorithms. Since these algorithms, such as regression and factor analysis, are readily available in packages,

---

*Quest is discussed in the present tense although it is only in the design stage.

Quest merely enables the preparation of data to be processed by these algorithms.

These language capabilities and design considerations are consistent with the thinking of the group considering "Entry and Query Language Design" in Baum [1, Sec. 5, p. 7]. A work session of the Symposium on Computer-Centered Data Base Systems considered various aspects of query language design and prepared a cogent statement of the group's consensus as to the desirable aspects of query languages.

Status of Quest

The Quest language is partially specified, but has not been implemented. The language itself is described in the next section of this chapter. In addition, the last section contains a Backus Normal Form* approximation to the formal syntax of the Quest language. The Backus notation is used to assist in the removal of ambiguities in the language, thus facilitating eventual implementation.**

Clearly, a translator must be written in order to translate from Quest language to a target language. This target language is, or in turn must be assembled or translated into, a machine language. For instance, the translator could be an Algol or Fortran program, which translates from the Quest language to a symbolic machine language such as Map for the IBM 7094 or Compass for the CDC 3400. Or the translator may translate from Quest to Fortran or Algol, which would then require additional translation. This

---

*See the Revised Report on the Algorithmic Language Algol 60 [3], for application of the Backus metalinguistic formulae to Algol.

**Not all ambiguities have been removed. It is not warranted at this level of investigation. Additional work is necessary prior to actual implementation. For example, ambiguities exist with respect to input-output formats and subscripted variables.

choice depends, in part, on the efficiency of translating to Fortran or Algol.

Conceptually, Quest is hardware independent. However, the more fundamental problem of utilizing Quest exists in terms of its implementation. The implementation issue is largely one of the system, both hardware and data structure.

Data Structure

Quest is conceived of working with data sets stored on lists or as data matrices.* Documented files and lists are envisioned to instruct specific user programs as to the data formats. Thus, Quest does not require user expressions of format. The variables or properties in a file or on a list are specified by the documentation. Selection of the file or list to query is a different problem. Quest is concerned, at one level, with the selection of a data set or the differentiation between data sets. At a lower level, Quest is concerned with the selection of entities from a data set or the differentiation within data sets. Here, it is assumed that the appropriate data set for query is readily accessible by specifying name, location, and time dimensions. A more detailed treatment of data set or file selection is presented in Chapter 6.

The hardware used in storing data sets is dependent upon whether data are stored as a serial file or on a list. When dealing with serial files,

---

*See Part IV, Chapter 9 for a tutorial discussion of organizing data. A data matrix is organized as a serial file with the rows representing observations or entities. Columns represent characteristics or properties of an entity. For lists, the storage location of the next item is specified rather than implied. Part of the item is a pointer to the next item on the list. Data stored as lists facilitates the insertion and deletion of items.

external devices such as magnetic tape are most used. For lists, large direct access storage devices are used.

Structure of Quest

A Quest program ia a means of representing a question as a set of commands. A Quest program may consist of five types of statements. These are: 1) data set selection statements, 2) entity selection statements, 3) modification statements, 4) manipulation statements, and 5) output statements.

Illustration of Quest

A Quest program is illustrated here. Presented for each type of statement is: 1) an illustrative standard form, 2) an example statement, and 3) an explanation of the example. The example pertains to a file containing data on a household travel behavior study. The entities of this example file are households. The properties of the households are characteristics relating to travel behavior.

The problem is to select households owning two or more cars, having three or more drivers, and who made more than ten trips on the day prior to the interview. Then summarize the selected households that are located in the same quarter square mile areal units.

Data set selection statement:

1. FOR <data set>, LOCATION = <location dimension>, TIME = <time dimension>;

2. FOR HHTRAVEL, LOCATION = SKOKIE, TIME = 64;

---

*Similarly the basic statement of the Fable on-line language of Mitre Corporation's Adam System consists of a selection-part, an action-part, and an output-part. The action-part is capable of modifying and selecting entities, and the output-part has a manipulation capacity. See Baum [1, p. 3-117] for a brief description of Fable.

3. Select a file called HHTRAVEL with a location dimension of SKOKIE and a time dimension of 64.

Entity selection statement:

1. IF <Boolean expression>;*

2. IF CAR GT 1 AND NDRIVER GT 2 AND AUTO.TRIPS GT 10;

3. Select household entities having more than one automobile, more than two licensed drivers, and that made more than ten auto trips on the day prior to the interview.

Modification statement:

1. ADD (<property name> = <arithmetic expression>)
   DELETE (<property name>);

2. ADD (OTHER.TRIPS = TRIPS - AUTO.TRIPS);

3. For the selected entities, a new property is created and none are deleted.

Manipulation statement:

1. SORT ON (<property name> $\left\{\begin{matrix}\text{ASCENDING}\\ \text{DESCENDING}\end{matrix}\right\}$, <property name>
   $\left\{\begin{matrix}\text{ASCENDING}\\ \text{DESCENDING}\end{matrix}\right\}$ ..);
   SUM (<property name>, <property name>) ON (<property name>)
   $\left[\begin{matrix}\text{AND COUNT}\\ \text{<empty>}\end{matrix}\right]$;

2. SORT ON (QSECNO ASCENDING);
   SUM (CAR, NDRIVER, AUTO.TRIPS, OTHER.TRIPS) ON (QSECNO) AND COUNT;

3. For the selected household entities, sort in ascending order by quarter-section number. Sum the values for the indicated properties for each of the quarter-section numbers and count the number of selected household entities within each quarter-section.

Additional Modification Statement:

1. ADD (AVATRPS = AUTO.TRIPS/COUNT, AVOTRPS = OTHER.TRIPS/COUNT);

---

*The Quest translator generates a target program that allows entities satisfying the Boolean expression to be processed by subsequent statements. The entity section or conditional statement qualifies data entities for further processing.

2. The purpose of this additional modification is to calculate the average number of auto trips and the average number of non-auto trips for each quarter section. The AUTO.TRIPS and OTHER.TRIP identifiers now refer to the property values summed in the manipulation statement.

Output Statement:

1. [PRINT, TITLE (<format statement>) / SAVE, NAME (<file name>) / DISPLAY, TITLE (<format statement>)] , (<property name list> (<format statement>)); END

2. PRINT, TITLE (X(40), *HOUSEHOLD TRAVEL BEHAVIOR STUDY* / X(45), *SKOKIE, ILL. 1964* //, X(40), *SUMMARIZED QUARTER-SECTION STATISTICS* /, X(35), *SELECTED HOUSEHOLDS WITH HIGH TRIP-MAKING CHARACTERISTICS *// . . .), (CAR, NDRIVER, AUTO.TRIPS, AVATRPS, OTHER.TRIPS, AVOTRPS, COUNT, QSECNO (X(20), 3F(10), F(10,2), F(10), F(10,2), 2F(10))); END

3. The output statement specifies whether properties of the selected or created entities be printed, displayed, or saved as a new file. In the example output statement, printing of the newly created quarter-section entities is specified. The total number of automobiles, total number of licensed drivers, total number of auto trips via other travel mode, average number of trips by other mode, number of selected households in each quarter section zone, and the quarter section zone number, are all specified to be printed for each of the newly created quarter section areal unit entities.

   The format statement symbolism is similar to that of PL/I. However, if the user only specifies the property names a standard format is used.

A more typical and straight-forward query might be:

FOR HH TRAVEL; IF INCOME GE 10000; PRINT, TITLE (. . . . .),

(INCOME, CARS, NDRIVERS, AUTO.TRIPS . . .(X(20), 4F(10)); END

where only those statements that are necessary to provide the needed output are used. It is re-emphasized that the above is merely illustrative of the type of capability sought. The example does not exhaust the capabilities of Quest, nor should the example be construed as representing a finalized syntatical or sematical structure of Quest.

Level of Quest

Quest, as described here, is a reference language rather than a hardware language. As such, Quest is a guide for all hardware representations. This philosophy is similar to the three levels of Algol, namely, a reference language, a publication language, and several hardware representations.

Symbolism of Quest

The symbolism used in Quest is similar to that used in PL/I.* That is, the symbols for arithmetic, relational, and Boolean operators are the same as used in PL/I. Thus, Quest is a PL/I-like language rather than, say, an Algol-like language.

Quest is designed to express queries to spatially distributed data. Because of the hardware independence, the constraints upon Quest are few. Like Algol, Quest, if implemented on various computers, may be modified slightly creating the existence of various dialects. On the other hand, the difficulty with most existing user-oriented languages is that the language and systems were developed simultaneously. Systems limitations constrained the development of the languages. Quest is an attempt to design a language that is not systems constrained. However, in implementation, Quest will be constrained by both systems considerations and the data structure.

Quest Semantics

Presented here is a detailed description of the semantical structure of the Quest language. The formalisms for syntactically describing Quest

---

*PL/I (Programming Language I) is being developed by SHARE and IBM. It has previously been referred to as NPL (New Programming Language). See McCracken [3] and Radin and Rogoway [4] for brief descriptions.

is the Backus Normal Form notation. This is presented in the last section of this chapter. The following description concentrates on the meaning and usage of Quest statements. Each type of statement--data set selection, entity selection, modification, manipulation, and output--is described and its meaning discussed.

Data Set Selection Statement. The purpose of this statement is to enable the selection of data from the various kinds of data that are available within the system. The data set selection statement is dependent upon a data directory or index that is part of the system. Thus, the user must be aware of the index and select files that are contained therein, using the prescribed identifiers for properties, or completely describe files that are not contained in the directory.

Selection of files requires specification by name of the desired file. If data of the same name are compartmentalized into different files according to location and/or time, these dimensions must also be specified. On the other hand, if each file has a unique name it is not necessary to designate the location and time dimensions.

An important capability desired in any Quest implementation is a means of matching or linking entities from one file to entities of another file. Linkage is based upon matching identical values for specified properties. For instance, matching person entities of a personnel file to person entities of an organization file requires matching on the persons name, social security number, employee number, or other unique identifiers. The data set selection statement performs the matching of entities from two files.* Within the entity selection statement and remaining statements it then becomes necessary to relate the property name identifiers with the appropriate files. For example, the matching may be on the location coordinate of one file to the location coordinate of another, i.e., XY(FILE.1)EQ XY(FILE.2), or the match may be on the employee number for person entities of the personnel and organization files, i.e., EMPNO(PERS)EQ EMPNO(ORGAN).

With increasing use of faster and cheaper direct access storage, means of responding to the above problems will be different. Rather, data will be stored on direct access storage devices, with address links between entities in different files.

For example, analysis of household travel data may be greatly facilitated by chaining trip entities from the same household together and to the master or household entity. Figure 5-1 illustrates the master or household entities within direct access storage. The trip entities are separate from the household entities. Each household entity heads a chain containing a list of trips made by members of that household.

In Quest, the need is to use chain addressing as a way of getting to a subordinate segment record from the master segment record, of getting to the next associated subordinate, and finally of getting from the last subordinate segment record to the master segment. To accomplish linkage or chaining it is necessary in data definition to specifically identify those

---

*To reduce search time and to facilitate implementation it is reasonable to require that the files be sorted in the same order (i.e., sorted on the identifier to be used for matching). This may require a manipulation statement to perform the sorting, prior to the entity selection statement that performs the match.

HOUSEHOLD SEGMENT

ADDRESS	HH No.	TRIP	INCOME	FAMILY SIZE
n	310			
n + 1	311			
n + 2	312			

CHAIN
K + 2
K
K + 1

TRIP SEGMENT

ADDRESS	ORIGIN ZONE	DESTINATION ZONE	TRIP PURPOSE	TIME OF TRIP
K	09	10		
K + 1	09	41		
K + 2	10	41		
K + 3	10	42		
K + m	41	10		
K + m + 1	41	28		
K + m + 2	42	10		

CHAIN
K + 3
K + m + 1
K + m
K + m + 2
n
K + m + m'
n + 1

Figure 5-1 Household - Trip Chains

properties that contain pointers rather than data. The Quest translator must generate the necessary instructions to search chains when a property identifier is used that is also a subordinate segment. For example, when the trip property of the households are used in a Quest program, the trip segment is entered and searched.

A special type of data selection statement enables the reading of files that are not in the system's directory or index. Because these data are not defined by the system, the file must be described in the query. The following example illustrates the addition of new data:

FOR DATA = <file name>, LOCATION = <location dimension>, TIME = <time dimension> (<property name>, <property name>, <property name>, ... , (<format>));

FOR DATA = HH.TRAVEL, TIME = 66; (CITY, SCOORD, YCOORD, STRUCTURE.TYPE, NCAR, NPER, . . . (A(1), 2F(5), 2I(1), I(2), . . .));

DATA is a reserved identifier, which when used as the first identifier in a selection statement, calls a read procedure. The read procedure reads from a system defined input device according to the format specified. Data input in this manner can be queried just as existing data in the system

Entity Selection Statement. This statement consists of conditional statements of the type:

1. IF <Boolean expression>;
2. IF <Boolean expression> THEN <unconditional statement>

    ELSE <conditional statement>;

The first type of entity selection statement qualifies entities for further processing. If the expression is false, for a particular entity, that entity is dropped and a new entity is read from the file and tested. The second selection statement enables the embedding of IF statements to test an entity for various conditions. The Boolean expressions of the embedded IF clauses are evaluated one after another until one yielding the value <u>true</u> is found. Then the next entity is evaluated. Upon testing all entities the statement following the compound entity selection statement is executed.

In dealing with spatially distributed data the determination of whether an entity lies within a spatial area is an important capability. When the locational property of data entities are geographic coordinates a procedure must be readily available to test whether the coordinates are within a polygon representing the spatial area. Thus, the Quest translator must have a built-in point-in-polygon procedure and a means of calling it. This call is of the form:

1. ADD (POLYGON, n, xp_1, yp_1, xp_2, yp_2, . . ., xp_n, yp_n);
2. IF <x-coordinate identifier> AND <y-coordinate identifier> IN

   [POLYGON OR ONLINE
<empty>];
3. ADD (<query area identifier> = <polygon number>);

The above sequence of statements implies the translator has some very specific capabilities. The testing of whether a point is within a polygon is a common part of spatial queries. Thus, the system should enable this test to be made with ease. For instance, the translator must have the following specialized capabilities:

1. An ADD (POLYGON . .) modification statement positioned prior to the entity selection statement designates new data that are to be added to core for comparison to each of the entities as they are tested.
2. POLYGON, is a reserved identifier designating that the subsequent values denote an array of vertice points of the polygon representing

the query area. The first value following POLYCON designates the length of the array. Following is the x-value for the first point, the y-value for the first point, the x-value for the second point, the y-value for the second point, and so on. The procedure can assign the $xp_{n+1} = xp_1$ and $yp_{n+1} = yp_1$ to close the polygon.*

3. The entity selection statement used for the polygon test also has a specialized form. The first two identifiers must be the property name assigned to the x-coordinate and y-coordinate, respectively. IN is a reserved operator, within the entity selection statement, which calls the polygon test procedure. OR ONLINE is an optional portion of the test. If OR ONLINE is included in the entity selection statement, and a coordinate point for an entity falls <u>on</u> a line segment of the polygon, the entity is called within the polygon and passes the test. If the coordinate point of an entity falls on a line segment of the polygon and the OR ONLINE is <u>not</u> part of the entity selection statement, that entity fails the test and is outside the polygon. Thus, the user has the option of including or excluding points falling on the edge of query areas.
4. Provision is made to add identifiers to each of the entities that fall within a polygon. This merely takes a modification

*Utilization of Quest in an on-line environment would be facilitated by adding polygon descriptions by means of a cathode ray tubelight pen device. Another reserved identifier would be necessary to instruct the computer to look to the display device for the polygon description.

statement following the entity selection. A polygon identification number is added to each entity passing the polygon test.

Often it is desired to partition spatially distributed data into a set of mutually exclusive areal units. This is accomplished by means of:

ADD (POLYGON, n, xp_1, yp_1, . . ., xp_n, yp_n);

IF X AND Y IN POLYGON THEN

ADD (AREA = 01); ELSE

ADD (POLYGON, n, xp_1, yp_1, . . ., xp_n, yp_n);

IF X AND Y IN POLYGON THEN

ADD (AREA = 02); ELSE

•

•

•

•

ADD (POLYGON, n, xp_1, yp_1, . . ., xp_n, yp_n);

IF X AND Y IN POLYGON THEN

ADD (AREA = 21); ;

PRINT . . . .

The recursive capability of the IF statement in Quest enables an entity to be tested for inclusion within each polygon, until it is found to be situated within one. Then a value is assigned to the query area property name and the program transfer to the end of the loop containing the compound IF statement. A transfer back to the beginning of the compound IF statement is then effectuated to test the next entity for polygon inclusion.

Where the query areas are not mutually exclusive, groups of three statements to test for a single polygon are repeated. Each group sets up its own loop through the entire file.

Modification Statement. This statement enables the addition or deletion of properties for each of the entities. These new properties may be generated from any simple arithmetic statement. Variables within an arithmetic statement are existing properties of the entities, such that new variables for each of the entities are created from the existing values of properties for that entity. To create a new property for all entities of a file by use of the modification statement, a transformation of existing properties is necessary or a constant is set equal to the new property. If a constant is set equal to the new property it becomes the new value for each entity of the file.

In most instances the modification statement should directly follow the entity selection statement. This is because statements between the first entity selection statement and a manipulation statement are placed within a single loop. Thus, the entity selection, a modification statement and possibly an output instruction are within a single loop. Each entity passes through this loop. As will be shown, all other statements generate their own loop to act upon the entire file. Having the modification statement directly following the entity selection statement saves looping through the entire file just to add or delete properties.*

All variables that are used in Quest are assumed to be of type floating point unless declared otherwise by a modification statement. For example:

---

*However, it may be necessary to create new properties before selecting entities, or after summing to create new entities. In these cases it is necessary to loop through an entire file to create new properties.

ADD (DECLARE <type>, <property name>, <property name>, . . .);

ADD (DECLARE INTEGER, STRUCTURE.TYPE, NCAR, NPER);

Declare is a reserved identifier which performs the conversion of variables to type integer or Boolean. Declare modification statements must be positioned first in a Quest program.

Manipulation Statement. This statement enables the sorting and summarization of data entities. Whereas the modification statement is restricted to action upon individual entities, the manipulation statement works with relationships between entities. Whereas the modifications statement performs the same operation on each entity, the manipulation statement changes the entity itself, by summarization to a new entity structure, or by reordering the entities.

Summarization is often used in the analysis of spatially distributed data. Data that are distributed over the spatial surface are aggregated to areal units. The aggregation reduces the number of entities and enables generalizations to be drawn about relative differences between areas. Summarization using Quest requires data be sorted on the property designating the summary unit. In this way, a summary unit is produced each time the key or property value changes. Each time the key changes, all the accumulated values are produced as the total value for the areal unit being summarized.

Suppose there exists, for each elementary school pupil in a large school district, a machine readable record. Each student is an entity and the machine records for all pupils constitute a file. Further suppose an automated street address translation system converted the home address of the students to geographic coordinates. With the existence of data of this type it is possible to allocate students to various configurations of elementary school enrollment areas by using Quest. Use of coordinated data, and polygons representing school enrollment areas, enables testing various school enrollment areas without having to recode or recollect data.

Using Quest, the pupil file is read, tested for inclusion within the polygons representing a test configuration of school enrollment areas, and the appropriate school enrollment area numbers are attached to the pupil entities.

```
FOR PUPIL, TIME = 66;
ADD (POLYGON, 6, 105, 210, 163, 210, 163, 255, 142, 255, 142, 230,
  105, 230);
IF XVALUE AND YVALUE IN POLYGON THEN
ADD (SCHOOL = LINCOLN); ELSE
ADD (POLYGON, n, xp1, yp1, xp2, yp2, . . ., xpn, ypn);
IF XVALUE AND YVALUE IN POLYGON THEN
ADD (SCHOOL = KENNEDY); ELSE
  . . .
  . . .
  . . .
```

This entity selection statement is used to assign pupils to appropriate categories for summarization. Then using a manipulation statement the pupil entities are sorted by school enrollment area number. All entities with identical enrollment area numbers are together, meaning, all the pupils assigned to a particular school are sequentially grouped in the resultant file. By means of another manipulation statement the pupil entities are summarized to school enrollment area entities. These new entities are output and analyzed for adequacy. If inadequate, a new configuration may be tested.

Assuming the level or grade of the students are denoted by numeric values within a single property, and that analysis requires the number of students in each grade, for each school enrollment area be known, the following entity selection statement must precede manipulation.

```
ADD (KG = 0, GR1 = 0, GR2 = 0, . . ., GR6 = 0)
IF GRADE EQ 0 THEN ADD (KG = 1); ELSE
IF GRADE EQ 1 THEN ADD (GR1 = 1); ELSE
IF GRADE EQ 2 THEN ADD (GR2 = 1); ELSE
   .     .
   .     .
IF GRADE EQ 6 THEN ADD (GR6 = 1);;
SORT ON (SCHOOL ASCENDING);
SUM (KG, GR1, GR2, . . ., GR6) ON (SCHOOL);
```

Output Statement. This statement outputs the answer to the query. By use of the output statement the user has control over the form of the query response.

There are many desirable features that could be built into the output statement. For example, automatic centering of headings would be convenient, as would automatic positioning of the property names of the print output statement as columnar headings. This latter could be facilitated by modification of the property names as follows:

```
ADD (KINDERGARTEN = KG, GRADE.1 = GR1, . . ., GRADE.6 = GR6);
PRINT, TITLE (. . .), (SCHOOL, KINDERGARTEN, GRADE.1, . . .,
GRADE.6 (X(15), A(15), 7F(12))); END
```

where the line of output following the heading would be the columnar headings positioned according to the format portion of the statement.

Another output option enables the user to specify an answer without concern for the presentation format. If a format statement is not provided the property list is automatically assigned to 12 column fields.

Finally, another output option enables the user to specify a graphic display, with symbols representing the presence of specified phenomena positioned in relation to the actual ground position of the phenomena. Different kinds of symbols are used to differentiate various value levels.

This option is activated by the reserved identifier, DISPLAY. This identifier initiates a plot of all the qualifying entities, according to the positional coordinates. Next the argument associated with the SCALE identifier gives the number of coordinate values per inch and the coordinates for the origin of the output. Unless otherwise specified by the RANGE identifier the values being displayed are assigned to the five equal classes, with a standard symbol for each class.

An output statement under the DISPLAY option is of the type:*

```
DISPLAY, TITLE (*HOME LOCATION OF FIRST*/*AND SECOND GRADE
STUDENTS*//* + = FIRST GRADER*/* 2 = SECOND GRADER*),
SCALE  (HOR = 10, VERT = 20) , ORIGIN (0, 0);

IF GR1 EQ 1, THEN

PLOT ( +, XVALUE, YVALUE) ELSE

IF GR2 EQ 1 THEN

PLOT (2, XVALUE, YVALUE);
```

where PLOT is a reserved identifier with the following arguments:

```
PLOT (<symbol>, <x-coordinate identifier>, <y-coordinate
identifier>, <property names>);
```

<symbol> is to specify the symbol. ER is reserved to signify standard symbols to be used for the five equal ranges.

---

*Acting on the school enrollment data, prior to aggregation.

The lowest range begins with the minimum value of the data to be plotted and the high range ends with the maximum value in the data array, i.e., (maximum-minimum)/5.

EN is reserved to signify that there is to be an equal number of observations in each of the five ranges. Again the maximum and minimum is determined from the data. ERO and ENO are the same as ER and EN, respectively, except the minimum is zero and the maximum is 100.

<x-coordinate identifier> is the property name of the x-coordinate.

<y-coordinate identifier> is the property name of the y-coordinate.

<property name> is the variable to be plotted.

Sample PLOT statements are of the form:

PLOT (X, XVALUE, YVALUE)

will plot an X at the home location of all pupils remaining after entity selection.

PLOT (ER, X, Y, GR1)

will plot a standard symbol at school locations that correspond to the number of first graders in attendance. *

PLOT statements are associated with each DISPLAY statement. If the PLOT statement immediately follows the DISPLAY statement, it is in the same operation loop through the entities. Intervening statements, such as an IF statement, imply additional manipulation is required.

Quest Logic

The sophisticated user desires to know how a Quest program is translated. Knowing this, the user is able to write more efficient Quest programs.

---

*Acting on the school enrollment data, after aggregation.

Similarly, translator programmers need to know the intent of the language designer in order to implement the language.

As has been shown, it is not necessary for the Quest programmer to devise loops for the programs to iterate through the entire data files. Rather, loops are automatically generated by Quest. Elimination of looping, as far as the programmer is concerned, is the greatest simplification provided by Quest. Yet, Quest programmers should be aware of how loops are generated in the Quest program. This section is concerned with the logic of programs generated by Quest programs.

Essentially, each Quest statement generates a do loop to iterate through the entire data file, performing just the action specified by that statement. Exceptions to this are DECLARE and POLYGON modification statements, and statements that are between the first entity selection statement and the end of the program or a manipulation statement, if one exists. Statements falling in this latter category lie within the do loop established by the entity selection statement. Figure 5-2 is a block diagram representing the logic of scanning statements in a Quest program by the translator. Figure 5-2 is generalized to illustrate the actions upon data, i.e., whether a statement establishes a loop or not.

<u>Backus Notation for Quest</u>

Statement Level

<query> ::= <program>

<program> ::= <declare modification statement><input statement><compound statement><output statement>END | <input statement><compound statement><output statement>END

<declare modification statement> ::= ADD (DECLARE INTEGER<property name list>);

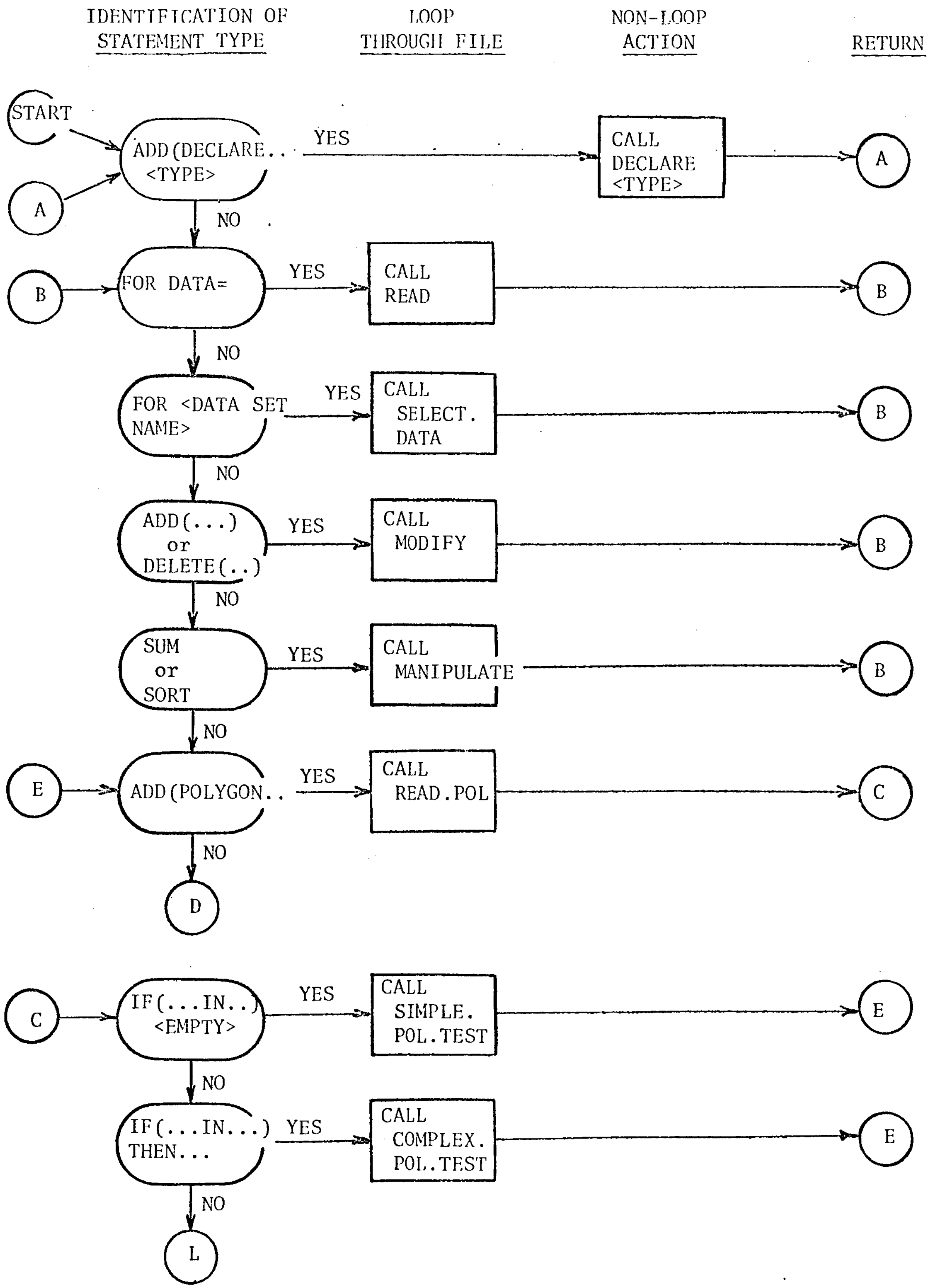


Figure 5-2. Quest Translator Logic (page 1 of 3).

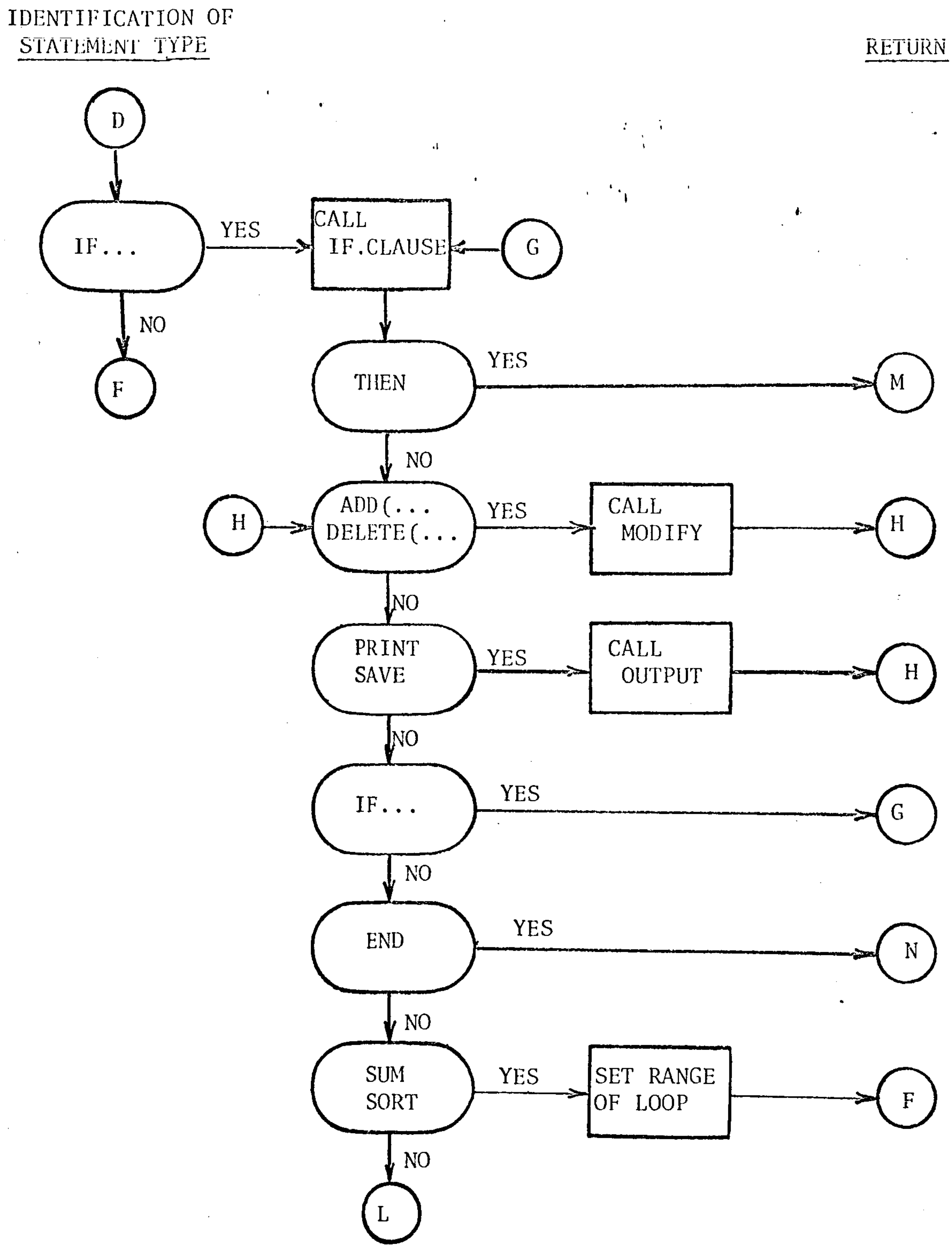


Figure 5-2. Quest Translator Logic (page 2 of 3).

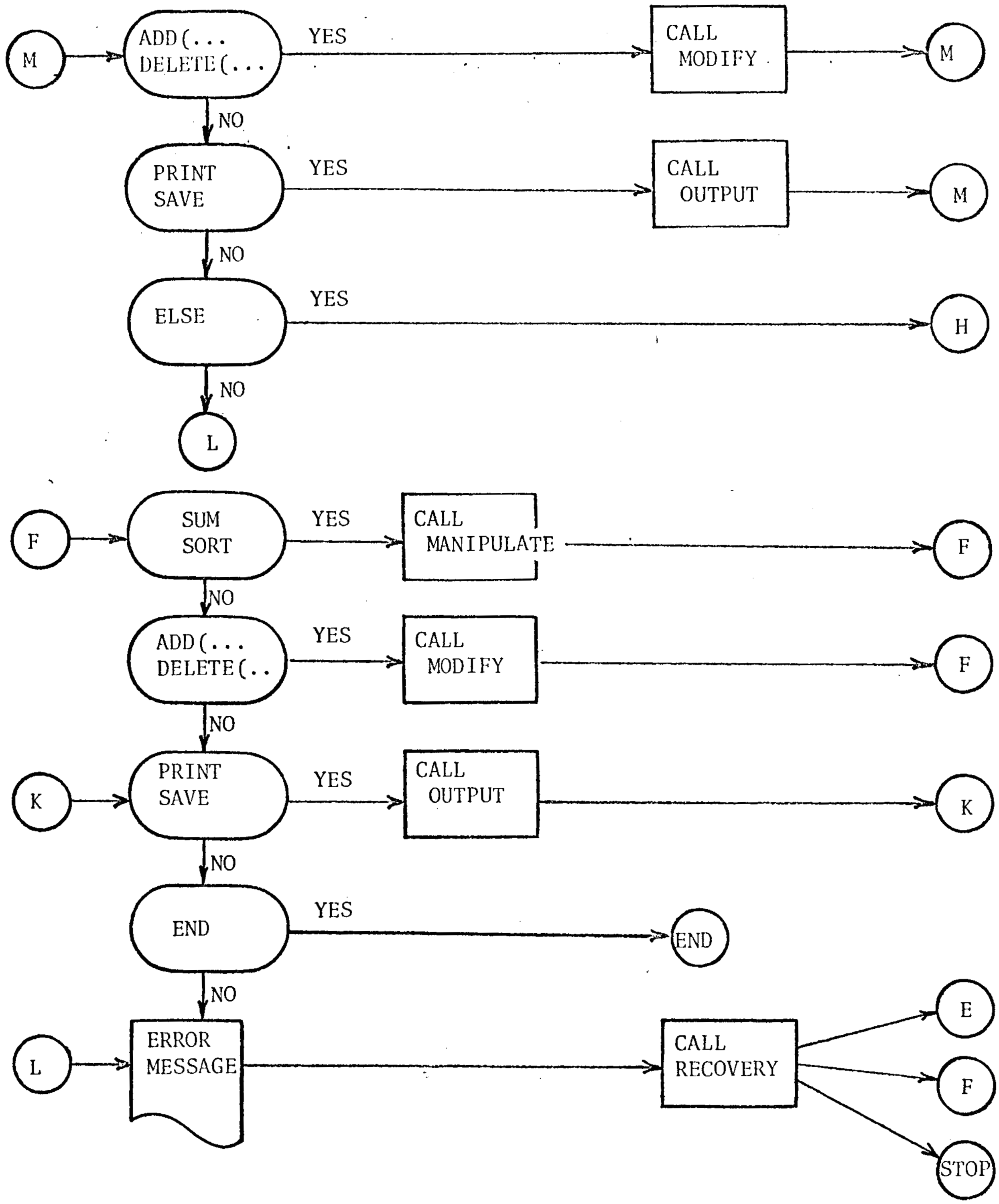


Figure 5-2. Quest Translator Logic (page 3 of 3).

| ADD(DECLARE LOGICAL<property name list>);

<input statement> ::= <data set selection statement> | FOR DATA= <file name>, <file dimensions>(<property name list>(<format statement>);

<statement> ::= <compound unconditional statement> | <entity selection statement>

<compound unconditional statement> ::= <unconditional statement> | <unconditional statement><compound unconditional statement>

<unconditional statement> ::= <input statement> | <modification statement> | <manipulation statement> | <output statement>

<data set selection statement> ::= <selection clause>; | <selection clause> AND<selection clause>;

<selection clause> ::= FOR<file name> | FOR<file name>,<file dimensions>

<entity selection statement> ::= <if statement> | <if in polygon statement>

<modification statement> ::= ADD(<assignment list>); DELETE(<property name list>); | <declare modification statement> | ADD(POLYGON, <polygon description>);

<manipulation statement> ::= SORT ON(<property-direction list>); | SUM (<property name list>)ON(<property name list>); | SUM(<property name list>)ON(<property name list>)AND COUNT;

<output statement> ::= PRINT, TITLE(<title format>),(<property name list> (<format statement>)); | SAVE, NAME(<file name>),(<property name list> (<format statement>)); | DISPLAY, TITLE(<title format>), SCALE(<abscissa scale>,<ordinate scale>), ORIGIN(<abscissa origin>,<ordinate origin>), <plot options>;

<plot options> ::= PLOT(<plot symbols>,<x-coordinate identifiers>,<y-coordinate identifiers>,<plot variable>)|<if statement> PLOT(<plot symbols>,<x-coordinate identifiers>,<y-coordinate identifier>,<plot variable>)

<if statement> ::= <if clause>; | <if clause>THEN<compound unconditional statement>ELSE<statement>;

<if clause> ::= IF<boolean expression>

<if in polygon statement> ::= <if in polygon clause>; | <if in polygon clause>THEN<compound unconditional statement>ELSE<statement>;

<if in polygon clause> ::= IF<x-coordinate identifier>AND<y-coordinate identifier>IN POLYGON OR ON LINE | IF<x-coordinate identifier>AND <y-coordinate identifier>IN POLYGON

<format statement> is not specified at this time

List and Name Level

<property name list> ::= <property name> | <property name>,<property name list>

<property name> ::= <identifier>

<property-direction list> ::= <property name><direction> | <property name> <direction>,<property-direction list>

<direction> ::= ASCENDING | DESCENDING

<identifier>* ::= <letter> | <identifier><letter> | <identifier><digit>** | <identifier>·<identifier>

<file name> ::= <identifier>

<file dimensions> ::= LOCATION=<location dimension>, TIME = <time dimension> |LOCATION = <location dimension> | TIME = <time dimension> | <empty>

<location dimension> ::= <identifier> | <value>,<value> | <value>

<time dimension> ::=<value string> | C <identifier>

---

*An identifier in Quest differs from Algol in that an open string is not permitted. Instead a period (.) is used.

**Defined in the Revised Report on the Algorithmic Language Algol 60 [2].

<value string> ::= <value> | <value string,value>

<assignment list> ::= <assignment statement> | <assignment statement>, <assignment list>

<assignment statement> ::= <identifier> ≐ <simple arithmetic expression>*

<polygon description> ::= <number of points>,<x-coordinate of point 1>, <y-coordinate of point 1>,<x-coordinate of point 2, y-coordinate of point 2,..., x-coordinate of point n y-coordinate of point n>

<plot variable> ::= <property name> | <empty>

Symbols

<letters> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |

<digits> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

<special characters> ::= * | + | - | x | / | ; |

<adding operator> ::= + | - |

<multiplying operator> ::= x | / |

<relational operator> ::= LT | LE | EQ | GE | GT | NE |

<logical operator> ::= NOT | AND | OR |

<end of statement> ::= ;

<plot symbols> ::= <letter> | <number> | <special characters> |EN | ER | ENO | ENR |

---

*Defined in the Revised Report on the Algorithmic Language Algol 60 [2].

References

1. Baum, C. and L. Gorsuch (Eds.), Proceedings of the Second Symposium on Computer-Centered Data Base Systems, System Development Corporation, TM-2624/100/00, 1965 (AD 625417).

2. McCracken, D. D., "The New Programming Language," Datamation, July, 1964.

3. Naur, P. (Ed.), "Revised Report on the Algorithmic Language Algol 60," contained in: Baumann, R., et al, Introduction to Algol, Prentice-Hall, 1964.

4. Radin, G. and H. P. Rogoway, "NPL: Highlights of a New Programming Language," Comm of ACM, Vol. 8, No. 1 (January 1965), p.9.

CHAPTER 6. DIRECTORY OF SPATIAL-TEMPORAL DATA

Introduction

In the query and retrieval of spatially distributed data, there must exist an efficient means of selecting the appropriate data sets for search. This chapter is concerned to the development of the concepts for an automated index or directory that enables this selection of data sets. This development of a data directory draws upon the topics of the previous three chapters. The notion of location and time dimensions of data, as developed in Part II, Chapter 3, is used to partition data sets into spatial-temporal compartments.

Each type of spatially distributed data is organized into subsets; one subset for each spatial-temporal compartment. Spatial-temporal compartments are derived from partitioning the spatial surface and time into three-dimensional areas. There are four types of compartments, depending upon whether location and time measures are uniform or non-uniform. These types of compartments are called quadrants. These quadrants contain data having similar spatial-temporal characteristics. The data set statement of Quest is interpreted to determine from which quadrant to select the named set. The quadrant plays an important role because the same data may exist in more than one quadrant. For example, building permit data may be filed by municipality or may be filed by uniform spatial regions. As described below, these duplicate data would be filed separately, each related to a different quadrant of the directory.

The directory itself is a combination list structure, and addressable storage that is addressable by mapping function of location and time ($f(x, y, t)$). The directory locations point to the actual location, reel numbers, file numbers, etc. of data sets. The concept involved is that a query specifies the kind of data, the location and the time. By means of the directory system the appropriate set is selected.

First, the partitioning of data observations into spatial-temporal compartments is discussed. Then these compartments are classified into spatial-temporal quadrants. Finally, problems of accessing these quadrants are discussed.

Part of the purpose of this chapter is to introduce urbanists to the technical problems of accessing diverse types of spatial-temporal data. The directory system introduced here demonstrates the complexities and immensity of a system for the automatic accessing of data sets. Obviously, the automated process described below could be simplified as a non-automated directory for the selection of data sets. However, the discipline of specifying an accessing process for the computer results in a thorough analysis of the steps involved.

Spatial-Temporal Compartments

Assuming various kinds of data are collected for a large urban area, how should these data be organized? Presumably, each type of data could be organized into area-wide files.* This would result in large files,

*The terms data file and data set are used synonymously in this chapter.

several hundred thousand entities for assessor's data on land ownerships.* What about data of the same type, but observed at different times? Should they be in the same file? Also, data such as building permits are collected by each individual municipality. Should they be combined in a single file?

For a general-purpose query capability, data with spatial-temporal characteristics should be organized by using the principal dimensions introduced in Part II, Chapter 3. The phenomena dimension, or in essence the kind of data, is already used to separate and organize data into files. In addition, for accessing data by location and/or time it is proposed that data be organized into files according to spatial-temporal compartments. Figure 6-1 illustrates the compartmentalization of the location and time dimensions on uniform scales. In Figure 6-1 the compartment having an x-value of 2, a y-value of 1 and a time of 3 is shown. Figure 6-1 illustrates uniform compartments, i.e., the area is partitioned into a regular grid and the time scale consists of regular intervals.**

The size of the regular grids is not specified. They may be based on 100 of 1000 foot grids, quarter-square mile, a square mile, etc. The choice is left for later analysis. The time interval for the uniform spatial-temporal compartments is defined as a year. Annual time periods cover many urban data such as licenses and permits.***

---

*If the urban area consists of more than a single county the data may not be compatible.

**Irregular size compartments are discussed in the next section.

***Obviously, other time intervals can be used for data that do not conform to annual time intervals. These situations merely do not fit within the uniform spatial-temporal compartment definition.

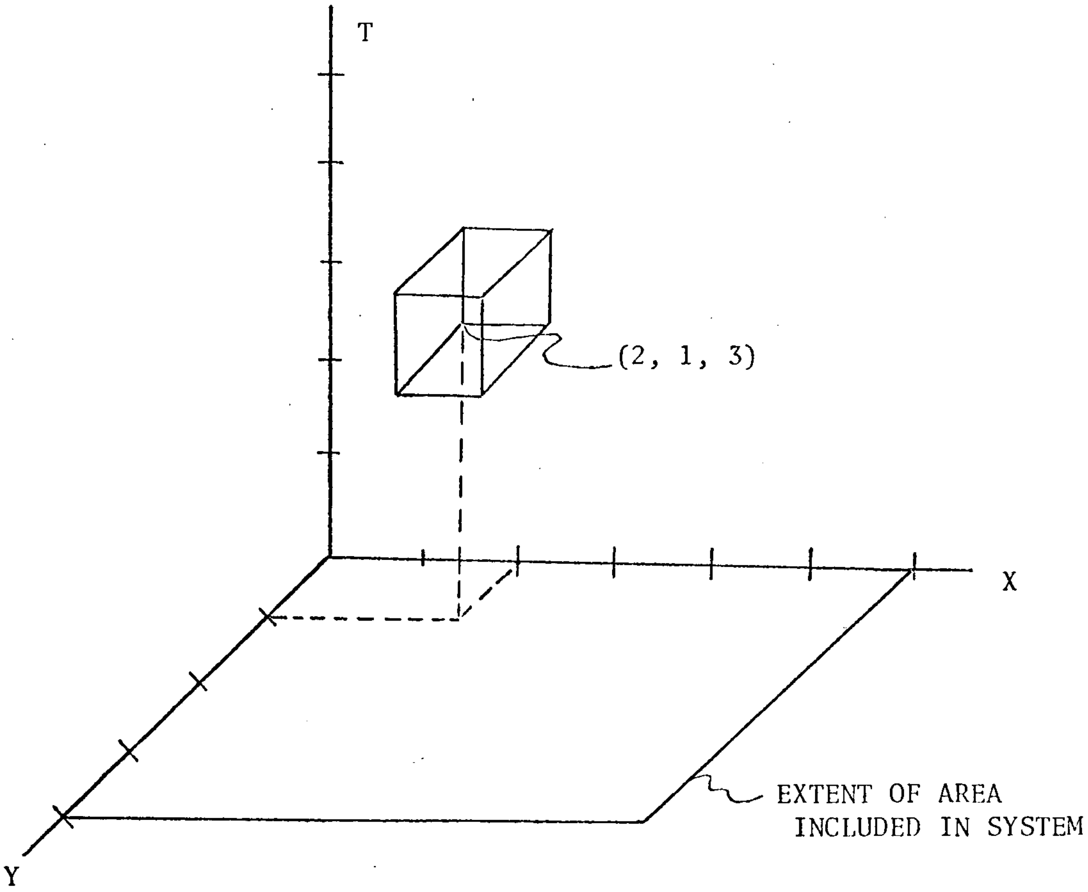


Figure 6-1 A Uniform Spatial-Temporal Compartment

Each kind of data, as distinguished by the name dimension, is allocated to several sets. The allocation of entities of the same kind is based upon the location and time dimensions. An entity is assigned to the spatial-temporal compartment that corresponds to its location and time dimensions. Thus, an entity that has locational coordinates that convert to spatial compartment (2,1) and a time of 3 years from the initial year is located in the spatial-temporal compartment (2, 1, 3) as shown in Figure 6-1. In other words, each type of data may consist of N_f files, where:

(1) $$N_f = X_{max}\, Y_{max}\, t_{present}$$

X_{max} = maximum x-value of the grid interval used to define spatial compartments

Y_{max} = maximum y-value of the grid interval used to define spatial compartments.

$t_{present}$ = present value on time scale measured in years from an origin, $t_o = 0$.

Data organized by spatial-temporal compartments offer great utility. A file for each spatial-temporal compartment enables one to specify location and time and then access just those data entities that are pertinent. In the retrieval of urban data, location and time are often elements of the selection criteria. In these cases, the distribution of data to files corresponding to spatial-temporal compartments enables the selection from those files that satisfy the specification of location and time. Means of specifying the spatial-temporal compartments are discussed in the following section.

Quadrants

Above, a uniform spatial-temporal compartment is defined. The adjective "uniform" restricts the location and time scales to be regular. For many data, uniform spatial-temporal compartments are inadequate. The data may be collected and utilized on the basis of arbitrary areal units rather than a grid system. Also, many data do not fit in the scheme of annual time intervals. Even data that are stored in files corresponding to uniform spatial-temporal compartments may also be stored in files corresponding to arbitrary areal units.*

Four kinds of spatial-temporal compartments are described here. First is the uniform spatial-temporal compartment. The others are time-varying compartments, uniform time-areal units and time-varying areal units. These are called type I, II, III, and IV quadrants, respectively. These four quadrants are defined by whether the spatial areas are regular or irregular and whether the time scale is annual or some other scale.** Table 6-1 summarizes the spatial-temporal variation for each quadrant.

Quadrant I

Figure 6-2 illustrates the organization of quadrant I data directory. The first quadrant is organized to be accessed by specifying the uniform measures of location and time. A function of location and time produces an address of a record containing the names of data types. Associated with these names are pointers to the beginning locations of the data files for compartment x, y, t.

To access the data designated by <file name>, in uniform spatial-temporal compartment x, y, t an address A_1 within quadrant I is calculated,

---

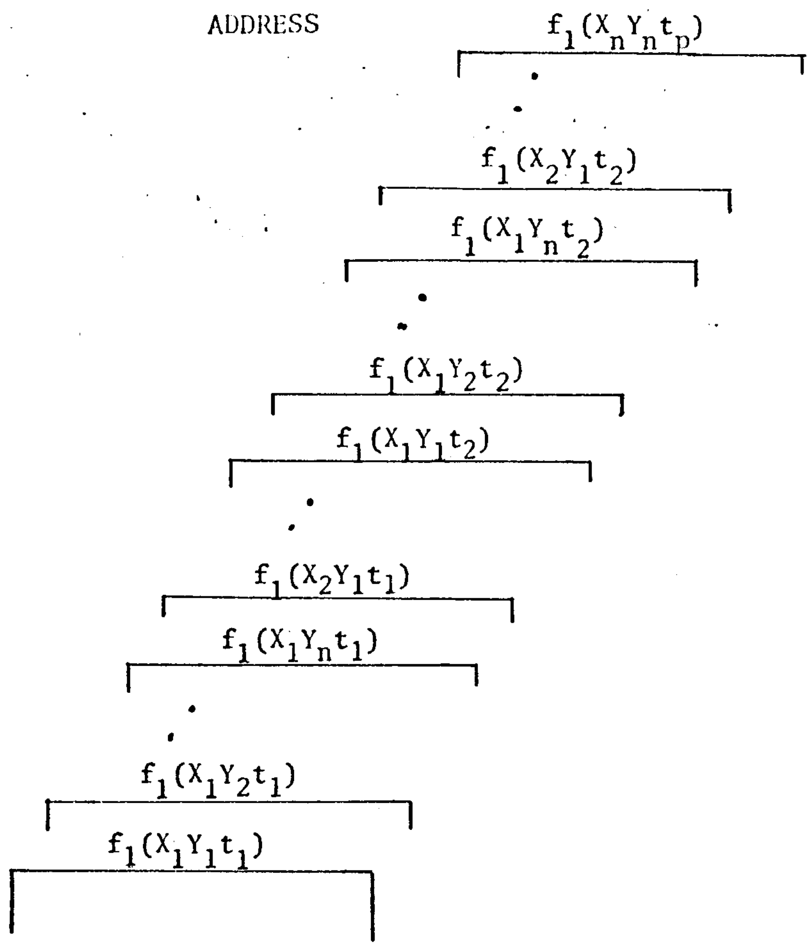
*This duplication of data within a system may be warranted if the particular kind of data is often accessed both by geographic coordinates and by areal units.

**The quadrant-spatial compartment terminology is analagous to the term "segments" as used in storage allocation for programs and data in computers for time sharing applications[1]. Segments are essentially files that can be accessed by calling segment descriptors.

TABLE 6-1

SPATIAL-TEMPORAL COMPARTMENTS

	Dimensions	
	Spatial	Temporal
Quadrant I	Uniform	Uniform
Quadrant II	Uniform	Variable
Quadrant III	Variable	Uniform
Quadrant IV	Variable	Variable



ADDRESS = $f_1(x, y, t)$

<FILE NAME 1>	POINTER
<FILE NAME 2>	POINTER
.	.
.	.
<FILE NAME nf>	POINTER

Figure 6-2 Quadrant I Data Directory: Uniform Spatial-Temporal Compartments

(2) $$A_1 = f_1(x, y, t)$$

The address is calculated from a mapping function of the location and time dimensions. Upon transfer to address A_1 the sequential record for x, y, t is scanned for the <file name i>. Associated with <file name i>, within the x, y, t entity, is a pointer to the beginning of the actual data of type i that are within spatial-temporal compartment x, y, t.

Quadrant II

Figure 6-3 illustrates the structure of quadrant II data directory. Quadrant II is a directory to data files having uniform spatial dimensions but time-varying time dimensions. Because the time dimension varies, the quadrant II directory is organized differently than the directory of quadrant I. In quadrant II, the directory is segmented by the type of data or the phenomena dimension. Within each segment a variable number of records may exist, one for each time period. Within each record, a mapping function of the location dimension enables the selection of the spatial compartment for that time period and data type.

To access data from quadrant II for <file name i>, spatial compartment x, y and time t, a file name list is searched for <file name i>. Associated with <file name i> on the list is the address K_{2i}. The address K_{2i} is an arbitrary point in quadrant II storage where the directory to spatial-temporal compartments of data type i begins. To select time t, a function of t is added to address K_{2i} to determine the location of the record for <file name i>, time t. Within this record, another mapping function is used to find the pointer to the data of spatial compartment x, y.

Assuming the pointer to data file location takes a word of core memory the transformation from two-dimensions to one within a record is:

(3) $$f(x, y) = y + x\,Y_{max}$$

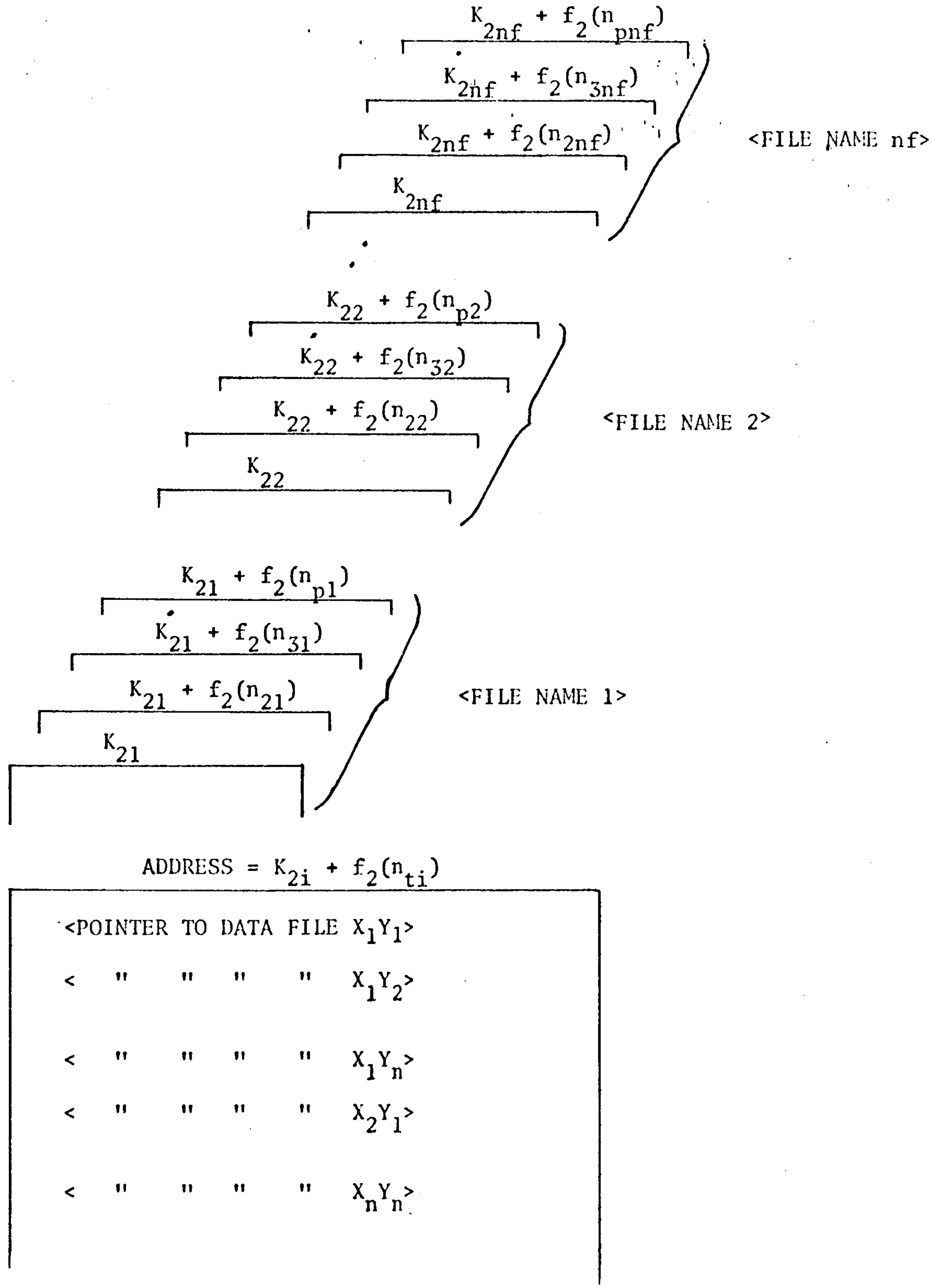


Figure 6-3 Quadrant II Data Directory:
Time-varying Spatial-Temporal Compartments

Where:

Y_{max} = maximum y-value for the spatial compartments.

Similarly, the mapping function for the address of the record itself is determined from the mapping function:

(4) $$K_{2i} + f_2\ (n_{ti}) = K_{2i} + (n_{ti} - 1)X_{max}Y_{max}$$

Where:

X_{max} = maximum x-value for the spatial compartments

n_{ti} = number of data sets of type i in quadrant II within the time interval $[t_o, t]$.*

Within the <file name i>, time t record, the mapping function, f(x, y) as defined in equation (3) is used to locate spatial compartment x, y. The desired address of the pointer to <file name i>, spatial-temporal compartment x, y, t in quadrant II is:

(5) $$K_{2i} + f_2\ (n_{ti}) + f(x, y) = K_{2i} + (n_{ti} - 1)X_{max}Y_{max} + y + xY_{max}$$

Quadrant III

Figure 6-4 illustrates quadrant III of the data directory. Quadrant III is designed to index data organized into files by non-uniform spatial areas and for annual time intervals. To handle the non-uniform areas, quadrant III is segmented by the arbitrary areal units. A list is used to access the beginning point of each areal unit index. The address for the

*Both t_o and t are included in the interval.

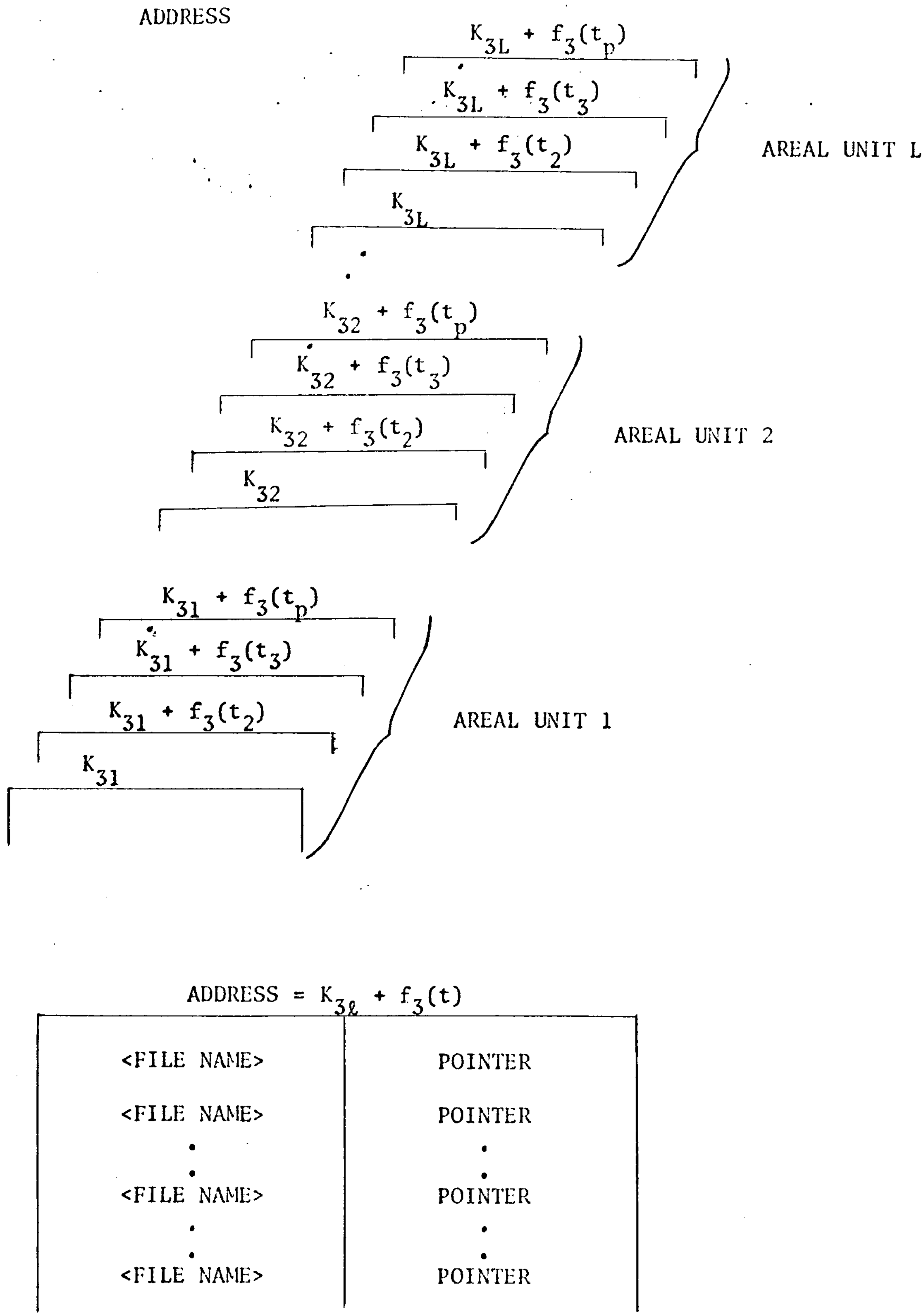


Figure 6-4 Quadrant III Data Directory:
Uniform Time-Areal Unit Compartments

beginning of areal unit ℓ index is represented as $K_{3\ell}$. Within the index for an areal unit a mapping function of time is used to select the appropriate record that corresponds to the specified time. Within the time record a list of file names are scanned to find a match for the file name specified. Upon finding a match the associated pointer directs one to the location of the actual data.

The access <file name i>, areal unit ℓ and time t the first step is to proceed through the areal unit list. Upon finding areal unit ℓ the associated address, $K_{3\ell}$, is accessed. Address $K_{3\ell}$ is the beginning of the index for the data of areal unit ℓ. To the address $K_{3\ell}$ a function of time t is added:

$$f_3(t) = (t - t_o)R_3 \tag{6}$$

Where:

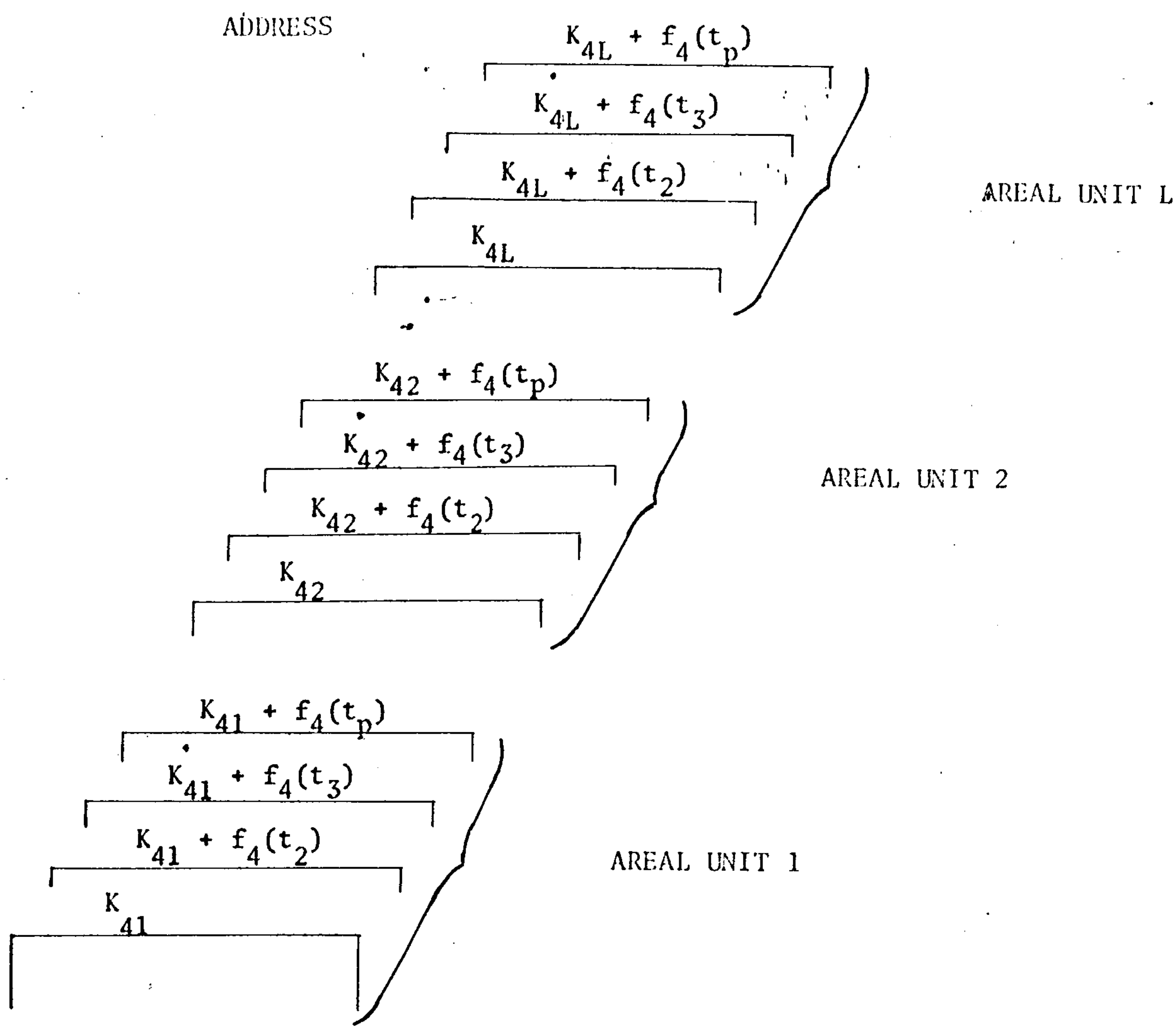
$t - t_o$ = number of years or time intervals from the initial observation to the specified time.

R_3 = a constant developed from the maximum number of data types in quadrant III times the number of words needed for file name and pointer.

Record $K_{3\ell} + f_3(t)$ is scanned for <file name i>. Upon finding <file name i> the associated pointer provides the location of the actual data for <file name i>, areal unit ℓ, and time t.

Quadrant IV

Figure 6-5 illustrates quadrant IV data directory. Quadrant IV is designed to index data organized into files by non-uniform spatial areas and for non-annual time intervals. Quadrant IV is organized similar to



ADDRESS = $K_{4i} + f_4(t)$

<FILE NAME>	POINTER
<FILE NAME>	POINTER
.	.
.	.
<FILE NAME>	POINTER
.	.
.	.
<FILE NAME>	POINTER

Figure 6-5 Quadrant IV Data Directory:
Time-varying Areal Unit Compartments

quadrant III except the time interval is variable. The new mapping function for time is:

(7) $$f_4(t) = n_{t\ell} R_4$$

Where:

$n_{t\ell}$ = number of data sets for areal unit ℓ within quadrant IV within the time interval $[t_o\ t]$.

R_4 = a constant developed from the maximum number of data types in quadrant IV times the number of words needed for the file name and pointer.

To access <file name i>, areal unit ℓ, and time t the first step is to proceed through the areal unit list. Upon finding areal unit ℓ the associated address, $K_{4\ell}$, is accessed. Address $K_{4\ell}$ is the beginning of the quadrant IV index, for areal unit ℓ. To address the data file corresponding to the specified time the function of time, $f_4(t)$ in equation (7) is added to $K_{4\ell}$. Record $K_{4\ell} + f_4(t)$ is scanned for <file name i>. Finding <file name i> the associated pointer provides the location of the actual data for <file name i>, areal unit ℓ, and time t.

Segment Selection

The selection of a file name segment in quadrant II and the selection of an areal unit segment in quadrant III or IV is accomplished using a chained list. Figure 6-6 represents the list for quadrant III.

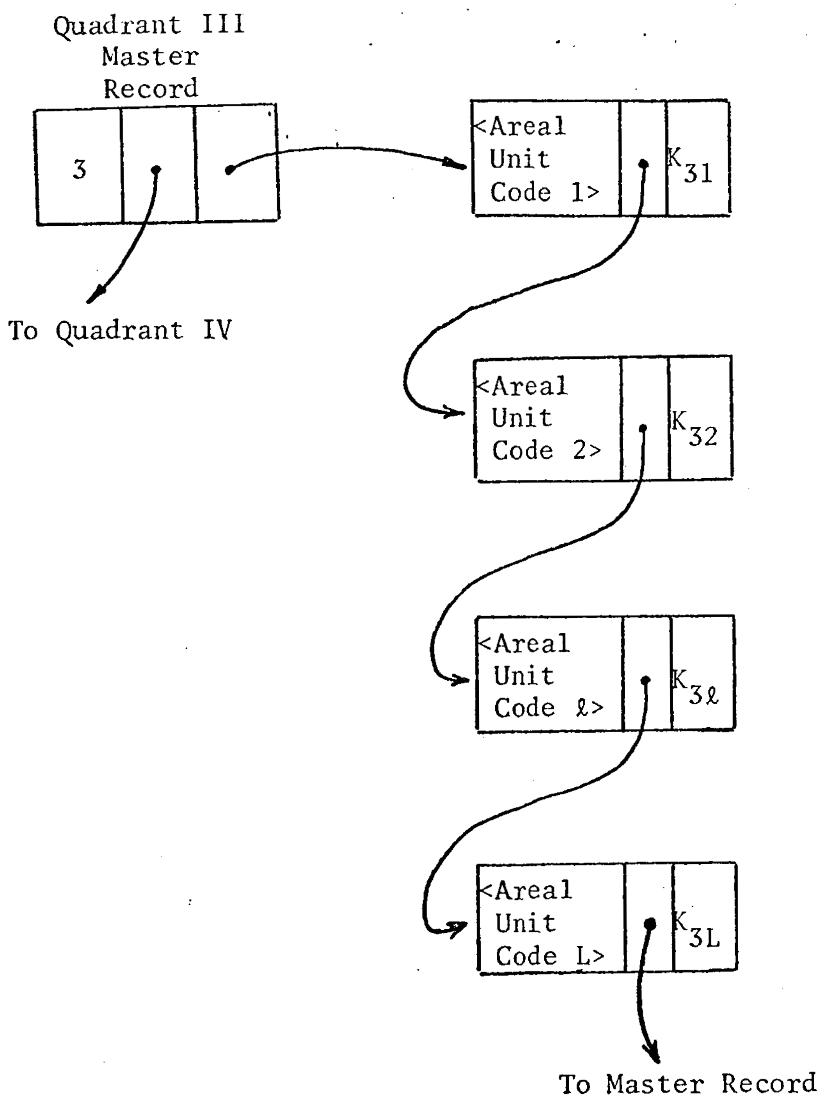


Figure 6-6. Segment Access List

Quadrant Selection

The next problem is to formulate an access to the directory or the selection of the appropriate quadrant. To describe accessing the directory, the process of selection from quadrants is presented. A narrative description of actions taken in interpreting the file selection statement facilitates explanation of the use of the data directory.

Data are organized in files according to their spatial-temporal compartments. Like compartments are grouped into the above described quadrants. In a query, location and time dimensions are specified to enable the selection of the required data file(s). From the specification of the location and time dimensions, a decision is made as to which quadrant contains a pointer to the appropriate file.

The data set selection statement of Quest, as described in Chapter 5, enables the specification of location and time dimensions. For example:

FOR <file name i>, LOCATION = SKOKIE, TIME = 66,4

FOR <file name i>, LOCATION = 14,8, TIME = C,8

These two data set selection statements illustrate some of the conventions used in Quest. In the first example, location is a code and time is specified as year and month. Additional values for time, such as the string 66, 4, 12, 16, 30.0 specifies year, month, day, hour, and minute. In the second statement, the two values for location are x and y values designating a spatial compartment. Time, in this second statement, is indicated as a code by the character C. The following value or character is the code representing a time value. The code must be converted to a time string of the type just described for selection of a quadrant and access to data.*

Both the location dimension and time dimension are determined to be in one of three states. The location dimension specifies a uniform spatial compartment, an areal unit, or is empty. Similarly, the time dimension specifies

*The conversion from time code to actual time value can be accomplished by a conversion table or a formula. A conversion table or formula is called upon for <file name i >.

an annual time interval, a non-annual time interval or is empty.

There are three choices each for the location dimension and the time dimension. The first concern is when both dimensions are non-empty. Of these, a direct decision is desired as to which quadrant to access. Table 6-1 represents the selection of quadrants from the choices of the location dimension and time dimension. A value of one in the spatial compartment column means the location dimension is an x and y value designating a spatial compartment.* When the location dimension denotes a spatial compartment and the time dimension denotes an annual time interval, quadrant I of the data directory is selected. Table 6-2 indicates the dimension combinations for selecting the other quadrants.

TABLE 6-2

SELECTION OF QUADRANT FROM LOCATION AND TIME DIMENSIONS

LOCATION DIMENSION			TIME DIMENSION			Octal Representation	Quadrant
Spatial Compartment	Areal Unit	Empty	Annual	Non-Annual	Empty		
1	0	0	1	0	0	44	I
1	0	0	0	1	0	42	II
0	1	0	1	0	0	24	III
0	1	0	0	1	0	22	IV

*To facilitate quadrant selection, representation of the location dimension state and the time dimension state as a single octal number facilitates subsequent tests. For example, view the six columns as binary numbers or two octal numbers. Interpreting the location dimension of the file selection statement, an octal 4 is assigned for values designating that the location dimension is a spatial compartment. Similarly, an octal 4 is assigned when the time dimension indicates an annual interval. Shift the location dimension octal number one position to the left. The octal 40 and octal 4 are combined using a masking operator to form the bit-by-bit logical product, resulting in an octal 44.

As stated above, the time dimension of the file selection statement may be empty. This is automatically interpreted as a request for the most current time value for the type of data being selected.

When the location dimension of a Quest file selection statement is empty, the initial interpretation is that, for a given time, all data files for spatial compartments are to be searched. Depending of the time dimension, quadrant I of II is accessed. This interpretation may be modified if a following statement of the Quest program is an ADD(POLYGON...) modification statement. Using the maximum and minimum coordinate values of the polygon a determination can be made to confine the search to a spatial compartment(s) in which the polygon is situated.

Data Directory Implementation

As suggested in the introduction to this discussion of a data directory, and as illustrated in the directory description, an automated directory is immense and complex. The obvious question is whether an automated directory is worthwhile. In answering this question the most important considerations relate to the frequency of using the directory and to the type of query system, i.e., an on-line query system versus a batch processing system.

An automated directory, such as described above can only be considered feasible if the frequency of using the directory warrants its permanent retention on direct access storage or if complex data structures require linkage on direct access storage. Secondly, an automated directory implies immediate need for query response. Conversely, a batch-process system does not require an automated directory. In batch-processing, narrative instruction suffices for the selection of data files to search. The user or data librarian consults a manual indexing system for the selection of appropriate data files.

An on-line data retrieval and management system enables users to handle real-time query needs. There must exist a sufficient need for immediate or real-time response for the retrieval and management of spatially distributed data to warrant an on-line system. It is in this context that an automated directory is considered.

An on-line system enables a dialogue between the user-programmer and computer system. Diagnostic assistance is built-into on-line programming systems. Similarly, a dialogue between the user-programmer and the computer system in the selection of data files is a necessary element of an automated data directory. Some of the safeguards that should be built-into the automated data directory are checks to insure that the specified file name exists in the data directory, that the location dimension of the data in the specified file is measured in the same units as the unit used in the file selection statement, and that the measurement units for the time dimension are the same in the data as are specified in the file selection statement. Also checks must be made to insure that only property names are used that are defined in the documented files.

As part of the on-line query and retrieval system and the automated data directory, it would be advisable to print out a format description of the data specified. Special emphasis in the data description should center on the form of the time and location dimensions to assist in the selection of a quadrant and the names of properties for use within the Quest program.

A more systematic discussion of computer systems for the storage and retrieval of spatial data is presented in the next chapter. The above statements concerning systems considerations are included here to emphasize the importance of the computer system in the development of a data directory. The computer systems requirements for implementation of data system capabilities conceived in this study are only implicit at this point. The next chapter explicitly considers the hardware requirements for the development of a data system. An important element of the hardware requirements relate to updating the data in the system and the directory.

For long term utility, a data directory must be updated. As well as designing the system to access data files, design consideration must be given to enable change and growth of both the system in its entirety and the data directory. Special utility programs must be written for periodic use in maintaining a current directory of the various spatial data.

In summary, the diverseness of spatial data of concern to urban analysts necessitates that data be organized in terms of their spatial and temporal characteristics. Data handling capabilities must exist for the storage, querying, retrieval and reporting of general classes of data. Up to now the concern has been with the conceptual problems of organizing and accessing data. Subsequent consideration will be with hardware and systems considerations for handling data.

References

1. Glaser, E. L., J. F. Couleur, and G. A. Oliver. System Design of a Computer for Time Sharing Applications. American Federation of Infromation Processing Societies 1965 Fall Joint Computer Conference. Spartan, 1965.

CHAPTER 7. HARDWARE-SOFTWARE SYSTEM CONFIGURATIONS

Introduction

Having presented various facets and concepts of general purpose systems for handling spatial data, system design and implementation considerations are now discussed. The approach is to consider a continuum of computer system configurations. These configurations range from a general purpose computer and a general purpose programming language to an exclusive and specialized computer with an on-line query capability. Between these two extremes are system configurations with limited specialized capabilities. The system characteristics and the advantages and disadvantages of these systems are discussed.

This discussion of system design and implementation is a narrative form of Figure 7-1. The cells of Figure 7-1 signify system considerations relating to the various system configurations and the various system phases. Figure 7-1, with empty cells merely provides an overview of the organization of this chapter.

Continuum of Hardware-Software Configurations

There are numerous possible hardware-software configurations that can be considered in the design of a system for handling spatial data. Rather than investigating all possibilities, five systems ranging from a very general configuration to a very specialized configuration are considered. It is felt that these configurations provide substantial insight as to the requirements

Continuum of Hardware-Software Configurations / SYSTEM PHASES	General Purpose Computer And General Purpose Programming Language (Level 1)	General Purpose Computer With Exclusive Auxillary Storage And Problem-Oriented Programming Language (Level 2)	Time-Shared, On-Line Computer And Problem Oriented Programming Language (Level 3)	On-Line Computer Special Query Language (Exclusive Use When Needed) (Level 4)	Specifically Designed, On-Line, Exclusive Use Computer, Special On-Line Query Language, And On-Line CRT. (Level 5)
DESIGN					
IMPLEMENTATION					
USE					
UPDATING					
ADVANTAGES					
DISADVANTAGES					

Figure 7-1 System Characteristics Overview

and capabilities of systems for handling spatial data.

At the first level, a system consisting of a general purpose computer and a general purpose programming language is considered. At this systems level, work is performed by batch processing. Size and speed of the processor is of less concern than rapid input-output capabilities. At this level, there is no requirement for exclusive use of any hardware. Variable amounts of computer time are used as needed.

The system of level 2 provides greater capability and ease of use. Level 2 consists of a general purpose computer with an exclusive direct access (disk or drum) storage and a problem-oriented programming language. The exclusive direct access storage enables accessing data, at various times and during different runs without having to load these data prior to each use and unload after each use. Also, level 2 assumes the implementation of a problem-oriented or task-oriented programming language. Problem-oriented programming languages are highly specialized to specific problems. Whether these are a programming language <u>per se</u> or a generalized package program, problem or task oriented systems are designed to handle a class of problems. The systems are designed to simplify problems within its class. Problems beyond its class cannot be or are not easily accommodated. Computer systems at level 2 enable manipulation of data with relative ease for given types of problems, such as file manipulation, data retrieval and/or data reporting. In addition data that are accessed often, are retained for continued usage.

Level 3 consists of on-line, time shared systems. Time-sharing permits numerous remote users to work directly (or on-line) with the same computer, concurrently, yet independently, [5, p.1]. With a file-oriented and problem-oriented programming language, such a system accommodates queries to exclusive direct access storage. Systems at level 3 are suitable for responding to queries but not for large data processing jobs.

Level 4 consists of a special purpose on-line computer system. These systems are special putpose with respect to the data base and with respect to the query language. Queries are to a common base of spatial data and the query language deals with a specific set of problems. When not used for on-line queries these systems may be operated as a batch processor for related large-scale data processing or for general data processing. Thus, a general purpose programming language must also be available for use in this mode. Stated in reverse, operation in batch processing mode is subject to interrupt for on-line queries.

Finally, level 5 consists of specially designed hardware and software, and on-line Cathode Ray Tube (CRT) devices. Systems at this level are designed to handle specified jobs and to provide desired capabilities. The specially designed features might consist of such features as digitizing coordinate locations using cathode ray tube and light pen, hardware for determining whether points are in polygons, and other devices to facilitate the manipulation of urban data. Although systems at this level do not exist and it is mere conjecture as to these configuration and features, it is assumed that systems of this level have all of the capabilities of the lower level plus specially designed features for handling spatial data, as well as additional built-in features.

System Phases

Specific phases of system development are considered in evaluating the characteristics of systems at each level along the continuum. These phases are design, implementation, use and updating. Each of these phases imposes constraints upon a system configuration.

System Design

The design of systems consists of defining the problem, selecting objectives, synthesizing or inventing alternative systems, analysis of the systems, and selection of the "best" system [4, p.9].

Problem definition is the transformation of an indeterminate situation into a pattern of factual data for the purpose of setting objectives. Expressing a problem is expressing an unsatisfied need. Thus, problem definition is determining needs. Of interest here are the needs of an agency for spatial data.

The product of problem definition is a set of objectives. Selecting or choosing objectives guide the search for alternative systems, and provide the criteria for selecting a system. Objectives describe the desired physical system in terms of things that are valued. This value system provides the means for judging the relative merits of the alternative physical systems to be synthesized.

There is considerable interplay between the selection of objectives and "systems synthesis." Systems synthesis is the development of alternative solutions to a problem. Systems synthesis consists of defining boundary conditions, and inputs and outputs. Then one proceeds to determine the functions or operations that must be performed, and relates those functions within a system to provide the desired outputs.

Systems analysis is the opposite of systems synthesis. In systems analysis, one decomposes the alternative systems to determine all relevant sequences. These consequences are compared to the initial objectives, providing a feedback to systems synthesis and selection of objectives. Unfortunately, the decision-making procedure is not often straight-forward. When the consequences of alternative systems are uncertain, independent and involve different scales of value, there is no comprehensive procedure for making a decision. Except for pieces, such as maximizing profit or minimizing cost, the selection of alternatives is usually intuitive elimination and selection process.*

The systems design problem is bounded by the statement of objectives or requirements, as discussed above, by technological limitations, and a limit of time and resources. It is of little value to design a system that is beyond present technology, beyond time limits, or beyond cost limits. Thus, feasibility considerations may impose a constraint upon systems performance objectives and require their reformulation.

A system that responds to user needs is the primary objective in system design. In synthesizing alternative systems, the needs and objectives are translated into hardware configuration. This hardware configuration is constrained by financial capacity. Given an upper limit as to hardware, a software system is designed for the given hardware configuration. The software design attempts to meet the needs and objectives. If a software system, for a given hardware configuration, is unable to meet the needs, the objectives and hardware are reconsidered. Either the objectives are scaled down or the hardware configuration is augumented, so that software can be designed to meet the objectives.

Implementation

The implementation phase of system development poses problems that influence system configuration. Important implementation considerations

---

*See Hall [4], Chapter 4 for a detailed treatment of this system design process. In his framework, this process is Exploratory Planning.

relate to staging the development of system components to maximize use at each stage, to minimize overall cost, and to reduce fluctuation of resource requirements. System performance and cost considerations also includes the training of personnel, development time for software, and lead times and arrival of hardware. Finally, a plan for collecting and organizing data is devised and implemented. On one hand, implementation is constrained by hardware and software development and installation schedules. On the other hand, system implementation is dependent upon accumulating the necessary data content. Resource allocation obviously plays an important role in the development of a computer system and a data base.

Use

Ease with which a system may be used is an important consideration in system design. As might be expected, systems which are easy to use perform more functions internally. If these internal functions are automated, there is a need for more powerful hardware and/or more sophisticated software. For instance an automated question-answering system requires more power and sophistication than a system without this capability. On the other hand, simpler computer systems may have considerable utility if coupled with skilled technicians. In a system of this type the computer system is a processor. The technicians provide a decision function that translates questions to queries, selects data for processing, specifies parameters to package programs, and organizes output. More complex computer systems are necessary to reduce the need for scarce skilled systems technicians and provide direct interface between the computer system and the users.

As the level increases on the continuum of hardware-software configurations, more decision functions are automated, and the easier it is for users to communicate directly with the computer system. As one moves along the continuum there is less need for special training to communicate with the computer system, or less need to go through a systems technician.

Updating Data

Ease of updating a data system is an important consideration of system design and in selecting a system configuration along the continuum. Methods for updating or adding to files that are retained on direct access storage are particularly important. As well as updating data, updating of data directories is necessary for systems containing diverse types of urban data.

System Characteristics

Some characteristics of systems along the continuum are presented below. The framework, developed above, is used to draw some generalizations and to discuss advantages and disadvantages of systems at the five levels along the continuum.

Level 1

Computer systems at level 1 are completely general purpose. The data system at this level is merely a set of package programs that process data stored on serial or tape files. As far as the computer system is concerned operation of the data system is just another job.

Systems at level 1 impose stringent design constraints. For the most part, systems at this level consist of package programs. These are utility or package programs that are submitted as individual runs. Utility programs for editing, formatting, search, and reporting are necessary. By specifying parameters, programs such as these have considerable power.

With the data system being just another user of the general purpose computer system the configuration of the data base is largely limited to serial or tape files that are suitable for batch processing and without permanent hardware requirements.

System implementation, use, and updating data are all controlled by the hardware-software configuration of a general purpose computer and a general purpose programming language. Implementation consists largely of writing package or utility programs and preparing data organized in serial files. System use consists of searching or processing data by specifying parameters and running package programs. In cases where the package programs cannot perform the job, special programs must be written using whatever general purpose programming language is available. Updating data at this level is reasonably straight forward. Updating consists of obtaining data and augmenting, replacing, or merging to existing data.

The chief advantage of systems at level 1 is that no fixed hardware cost is required. Using someone elses computer installation, the computer charges are variable, and depend on the amount of computer time used. The package programs that make up the software component of the system remain largely unautomated. Choice of data and query formulation and application are performed by a systems technician.

The disadvantages of systems at level 1 are primarily related to restrictions of batch-processing, serial files, and the limited range of package programs. Batch processing results in a time lag for responding to a query. This lag is essentially the turn-around time on the computer. To gain the advantage of no fixed hardware cost, the computer is shared with other users in a batch processing mode of operation. Turn-around time depends largely upon the queue length of jobs. Also related to batch processing and no hardware investment is the restraint upon data organization into serial files. Data organized in serial files may necessitate scanning an entire file for order to access a few entities, whereas if these data were retained on addressable storage the entities could be directly accessed. Finally, a package program is written to accomodate a class of problems. Problems beyond classes for which package programs exist require a special program be written. At this level special programs must be written in a general purpose programming language and may require considerable skill, effort, and time.

Systems at level 1, with problem-oriented language-like package programs, generally exhaust present systems for handling urban or spatial data. Because of undemonstrated or uncertain payoffs, development of urban data systems for handling spatial data for planning purposes have largely been forced into systems at level 1 where no fixed hardware cost is necessary. Until a payoff for urban data systems can be demonstrated, this will continue.

Level 2

The hardware-software configuration of systems at level 2 along the continuum offer greater specialized capability for handling spatial data. Use of a problem-oriented language facilitates expression of a query as a processing request in a form more convenient to the user. Secondly, data may be retained on exclusive addressable storage, thus facilitating accessing specific entities.

Computer systems at level 2 enable considerably more design freedom than is available at level 1. This is particularly due to the availability of direct access storage. Often used and accessed data can be structured and retained for continued usage.

Package programs and a problem-oriented language cannot always be distinguished as the so-called problem oriented language may be a means of specifying parameters stated in English-like sentences to package programs. Romtran [6] is a problem-oriented language used to specify parameters and the performance of arithmetic operations for individual package programs for graphic display of spatial data. Span [1] is a system of package programs whose parameters are specified by an English-like language. As discussed in Part I, Chapter 2, Span has a considerable range of capabilities from file manipulation to statistical analysis. The large number of available options or packages enables Span to perform as a language as far as most users are concerned. Span can handle the bulk of file management and file processing needs, for unit-record serial files. However, Span would have to be augmented to handle data stored on auxillary direct access storage to be used in systems at level 2.

Selection or design of a problem-oriented programming language is related to data structure. Many problem-oriented languages such as Span and Romtran can only handle unit-length records. Others such as Mark III [7] can handle variable length records. Software must be designed to handle the form of the data. Particularly important is the design of the data structure for the data to be retained on auxillary storage.

Implementation consists of designing the problem-oriented language and developing this software system, or installing an existing software system. Implementation also consists of preparing the data base, both the serial files and the direct access files. Use of systems at level 2 consists of running programs whose parameters are specified in a problem-oriented language. These programs process data stored in serial files or on the exclusive auxillary direct access storage. Updating the auxillary storage device poses a new, although not serious, problem. Updating is the opposite of retrieval. By means of a mapping function or scanning a list the appropriate slot is found. New data can be positioned or existing data can be retrieved.

The primary advantage of systems at level 2 is that a minimal hardware investment provides much greater capability to store complex data structures, and to access often used data without requiring sequential searching. In addition, a problem-oriented programming language, with a broad range of capability, enables the user himself, to perform much of the data manipulation, search, and processing.

The disadvantages of systems at level 2 again relate to the turn-around time on a general purpose computer operated as a batch-processor. Again, the data system is just another job to the computer. The second disadvantage is that updating the auxillary storage consumes computer time just as accessing it for retrieval purposes does.

Level 3

Level 3 hardware-software configurations are oriented to facilitate query response, but at the expense of large-scale data processing. Systems at this level enable rapid access to data in addressable storage. The on-line query formulation and immediate computer access via time-sharing provide a "real-time" query system. Real-time in the sense that query response is virtually immediate with respect to using the answer.

Design of systems at level 3 are clearly a response to a need for handling queries that are posed at irregular times and that require immediate answers. Such a need might be to inquire as to account balances, to determine owners of vehicles given the license number, to determine the present location of freight cars, or to determine the status of an order. Difficulties arise in designing file manipulation capabilities for a time-shared computer system.

Many time-shared systems are for general purpose scientific programming, using a common language. With "simultaneous" users, a time-sharing computer must converse in a single language. Thus, the user of the data system must converse or query the data using the language. Package programs or routines written in this common language could be called and parameters specified to negate a good share of this problem, however. An important design consideration is to protect data and programs from "simultaneous" users. Safeguards must be designed in the time-sharing system and/or the data system to insure that data and programs are not destroyed or accessed by others.

Implementation, use, and updating considerations are greatly influenced by the choice of a time-sharing system. Implementation consists of preparing and protecting the data base, and to write often used queries or elements of queries as package programs that are activated by specifying on a single entity. Entities are accessed by a mapping function that calculates a storage location from a key or are accessed using list structures. For example, given a license number or social security number the associated data entity can be accessed. Updating such a file is again the reverse of retrieving. The retrieval key is used to find a location for storing the data entity.

The primary advantage of systems at level 3 relate to the capability of immediate response to queries that are directed to addressable data. This advantage is of great importance to many data users. Immediacy of response is necessary for many functions, and an on-line query system is essential to provide this immediacy. In addition, time-sharing is advantageous to minimize the fixed hardware costs. Computer costs are primarily dependent upon usage.

Some disadvantages of systems at level 3 are: no choice of a programming language, the time-sharing systems may be implemented on hardware beyond the needs for file manipulation, problems in protecting data and programs, and large-scale processing or long sequential searches cannot be efficiently conducted in the time-sharing mode. Both the programming language and hardware of a time-sharing system are probably geared more for scientific computing than file maintenance and queries. This might be avoided if enough users having similar query needs were to band together and develop their own time-sharing system. This would probably prove expensive to develop but more efficient in operation. The problems of protecting data and programs from destruction by simultaneous users is related to the need for an exclusive random access storage device for use in the data system. Protection must be designed into the system so that other users cannot access the exclusive auxillary data. Probably the most serious disadvantage of level 3 systems relate to the exclusion of large-scale data processing or long searches. Meeting the objectives of immediate query response and minimizing hardware investment, sacrifices capabilities for production work. Production work must be done on a computer system that is not operated in a time-sharing mode. However, the programs may be written and debugged on the time-sharing system.

Level 4

Level 4 systems consist of a hardware-software configuration that can be operated in an on-line mode or as a batch processor. The on-line mode has priority and interrupts work being run in the batch processing modes: Level 4 systems also have exclusive auxillary addressable stroage and a query system amenable to posing queries by specifying parameters using remote consoles. Level 4 systems enable both queries and production. Subject to query interrupt, time may be leased to other users to reduce fixed hardware costs.

Design of systems at level 4 involve formulation of a method for changing mode of operation from a batch processor to an on-line query system and back again. Provision must be made to interrupt, to save what was being processed and to restore the original job upon completion of the query.

Implementation, use, and updating phases of level 4 systems are substantially free from major hardware-software constraints. Level 4 systems facilitate both query capability and large-scale data processing. In implementation the major concern is with developing and specialized monitor system that can operate in two modes. One mode being a production system and the other being a query system. Production or large-scale data processing typical of the needs of functional agencies would operate in the batch-processing mode. Work in this mode would be subject to interrupt to enable immediate response to queries to the data base. The form of the data base is extremely flexible. Level 4 systems are primarily to serve this data base. Its main function is in file manipulation and data retrieval for a single data base.

The chief advantage of level 4 systems relate to the versatility and flexibility of being able to respond to queries and to perform large-scale data processing functions. As well as the ability to do both, these systems give priority to queries. Systems at this level offer great power in providing a full range of data system capability. Whereas systems of lower levels may perform one function well they may have compromised their performance in another area. Level 4 systems poses a powerful enough hardware-software configuration to provide a full range of urban data system needs.

The chief disadvantage of level 4 systems is that it involves an expensive hardware-software configuration. To justify an expensive hardware-software configuration sufficient system payoff must be demonstrated. Yet there is too little experience with flexible data systems to demonstrate or determine the payoffs with respect to spatial or urban data systems. To justify a data system at level 4, one must have a need for frequent queries to a large data base that requires fast response, and a need for large scale processing of data. However, this large scale data processing need may be fulfilled by making the computer system available to others to use in the batch processing mode.

Level 5

Level 5 systems consist of a specially designed hardware and specially designed software. Specially designed, in that the procedures for handling the problems and needs of spatial data are designed into the system. For example, it would not be necessary to devise logic and write a program to determine whether data are within a set of areal units or to manipulate these data for summarizing to these areal units. Rather, these functions are designed into the hardware or software. Quest, described in Part II, Chapter 5, exemplifies in a limited way, this approach with its built-in-polygon test. Level 5 systems, as does level 4 systems, include two mode usage and exclusive direct access storage. In fact, the entire configuration of level 5 calls for exclusive use in a spatial data system because the specially designed features preclude use by others.

The design of level 5 systems assumes little in the way of hardware constraints. Functions that are often performed may warrant the design of hardware that will amortize the cost through repeated usage. Such functions as point-in-polygon tests, a directory for spatial data, and two dimensional storage for map or coordinate data, may warrant specially designed hardware. So in the design stage, there is not only the need to conceive software systems, but also to consider whether a function is better accomplished by designing a hardware component to perform the action of a complex or often used subprogram. At level 5 the designer is not working within a given hardware configuration; the design process includes design of hardware components as needed.

Two-dimensional devices for graphical display and graphical input are important elements of a system specifically for spatial data. Cathode Ray Tube (CRT) devices are essential for the efficient input-output capabilities. CRT's are an integral part of level 5 systems. Display of data that are positioned in two or three dimensional space and digitizing of spatial data may be facilitated through the use of specially designed CRT's that are on-line with level 5 systems.

Another important built-in feature of a level 5 system is a street address to locational coordinate translation system. Dial [3] has developed a software system that requires several stages of discrete operations. It is envisioned that a level 5 system performs these stages internally and without several independent computer submissions. Again, the design problem is to determine which functions are to be performed by hardware and which by software. Such a translation system needs to be flexible enough to accept addresses in various forms and order and still convert these to coordinate locations. To achieve this flexibility the translation system must consist of a capability to distinguish between street number, street name, street type, post office name, etc. The directory of street address ranges to coordinate locations may also be stored in direct access storage to enable processing of unordered addresses. This translation capability is an important task in handling urban data and is considered to be essential for systems at level 5.

Implementation of level 5 systems include the construction and testing of hardware components as well as developing a software system.

Implementation involves development of a query system and a data handling capability possessing a great deal of power with a minimal degree of user computer skills. One of the objectives in automating most functions is to bring the computer closer to the user.

Use of level 5 systems and data updating are highly automated. With the design and implementation emphasis on automating often used functions, either by means of specially designed hardware or software, much of the burden of posing queries or handling data is removed from the user. Desired user capabilities such as a simplified means of expressing on-line queries, and a user-oriented programming language for manipulating spatial data, facilitate system use by non-specialists in computer programming. As in systems of level 4, the systems of level 5 possess an interrupt for on-line queries. As is programming at this level, updating data is concerned to be done in a very automated way. Data directly from sources are added to the system in much the same way as regular reports are produced. Updating and reporting are both considered a normal function that does not require special action. These functions are built-in.

The advantages of level 5 systems relate to specially designed features which enable efficient performance at the desired functions. Systems whose hardware as well as software are designed for the explicit needs of handling spatial data provide a powerful user-oriented tool for manipulating and analysis of spatial data. The built-in capabilities, such as point-in-polygon tests, graphic display, and street address translation enables rapid and low unit cost response.

The disadvantages of level 5 systems relate to their high design and development cost. A considerable hardware and software investment is necessary prior to usage. Amortization of these costs necessitates that these systems be used extensively, that these systems provide a substantial savings over other systems or that systems of lessor power could not perform the desired objectives. Another disadvantage is that these systems require full utilization by the primary agency as the specially designed features probably preclude use of the system by other unrelated users. Thus, spare time is lost rather than sold to the others. The needs for the system must fully justify the costs of the system. This is an extremely difficult estimate in light of the dearth of experience and the difficulty in assigning value to information used for planning in the public sector.

Summary

The range of hardware-software configurations discussed here illustrate the interrelationships between the hardware and software. A given capability may be achieved by a limited hardware configuration and an extensive software system or it can be achieved by a more powerful hardware configuration and thereby require less extensive software. Trade-off consideration between hardware and software is an area that needs more study. The above presentation only illustrates the range and poses the problem in some sort of framework.

References

1. Almendinger, V. V., Span Reference Manual: Span Operation, System Development Corporation, TM-1563/010/01, February 1965 (AD 613 284).

2. Baum, C. and L. Gorsuch (Eds.), Proceedings of the Second Symposium on Computer-Centered Data Base Systems, System Development Corporation, TM-2624/100, 1965 (AD 625 417).

3. Dial, R. B., Street Address Translation System, Urban Data Center, University of Washington, 1964.

4. Hall, A. D., A Methodology for Systems Engineering, D. Van Nostrand, 1962.

5. Hodge, C. (Ed.), "Direct Dialing to Digital Computers," Bulletin of the Inter-University Communications Council (EDUCOM), Vol. 1, No. 1 January 1966.

6. Horwood, E. M., Using Computer Graphics in Community Renewal, CRP Guide No. 1, Urban Renewal Administration, Housing and Home Finance Agency, 1963 (Available from U.S. Government Print Office, Washington, D.C.)

7. Postley, J., "Mark III File Management System," Contained in Baum and Gorsuch [2].

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Department of Geography Northwestern University Evanston, Illinois.	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

SPATIAL DATA SYSTEMS: SYSTEMS CONSIDERATIONS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report

5. AUTHOR(S) *(Last name, first name, initial)*

Dueker, Kenneth J.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December, 1966		

8a. CONTRACT OR GRANT NO. Nonr 1228 (37) b. PROJECT NO. Task No. 389-143 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Report No. 5 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES

The distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Geography Branch Office of Naval Research

13. ABSTRACT

Spatial data systems are concerned with the organization, handling, and retrieval of data whose spatial position is of concern. Spatial data are of particular concern in urban and transportation planning. In these fields considerable attention is given to spatial and temporal variations of data.

The three volume report presents a discussion of concepts and techniques that are essential in moving towards flexible and responsive urban information systems. The following areas are emphasized:

1. Explication of terms associated with spatial data.
2. Discussion of means of organizing spatial data for flexible and efficient retrieval.
3. Investigation of data handling capabilities for organizing and manipulating spatial data.
4. Presentation of these topics in a tutorial form, conceivably to serve as a text where none presently exist.

The greater speeds and storage capacities of newer computers requires new concepts of data organization and new means to create and access these more complex data structures. Of particular concern in urban and transportation planning are needs to link separately collected data that relate to the same phenomena or spatial locations, and a need for user-oriented data handling capabilities. These needs are explored and recommendations are made.

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Geography						
Spatial Data Systems						
Urban Data						
Language						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through ________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through ________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through ________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

Unclassified

Security Classification